
A New Generative Adversarial Network for Improving Classification Performance for Imbalanced Data

Emilija Strelcenia (S5223247)



**A New Generative Adversarial Network for
Improving Classification Performance for
Imbalanced Data**

A thesis submitted in partial fulfilment for the
degree of Doctor of Philosophy
in the
Department of Creative Technology Bournemouth
University

September 2023

Abstract

Data is a common issue in many industries, particularly in fields such as fraud detection and medical diagnosis. Imbalanced data refers to datasets where the distribution of classes is not equal, resulting in an over-representation of one class and an under-representation of another. This can lead to biased and inaccurate machine learning models, as the algorithm may be inclined to favour the majority class and overlook important patterns in the minority class. Various sectors have utilised deep neural networks for data synthesis. However, according to research papers in these fields, balanced data outperforms imbalanced data when it comes to deep neural networks. Although deep generative approaches, such as Generative Adversarial Networks (GANs), are an efficient method of augmenting high-dimensional data, there is a lack of research on their effectiveness with credit card or breast cancer data and the current methods demonstrate limitations. Our research focuses on obtaining a great number of sets of data that are valid and resemble the minority class, in this case, fraudulent or malignant samples. Having more data like this can be used to train a binary classifier so it's effective against fraud or cancer diagnosis. To overcome challenges opposed to existing methods we have developed a novel GAN-based method called K-CGAN, which has been tested on credit card fraud and breast cancer data. K-CGAN is designed to generate synthetic data that resembles the minority class, effectively balancing the dataset and improving the performance of binary classifiers. Our research demonstrates the effectiveness of K-CGAN in handling complex data imbalance problems often encountered in practical applications. In addition, the experiments performed on different datasets indicate that K-CGAN can be used for various purposes. The application of machine learning algorithms in various industries has become increasingly popular in recent years. However, the quality and quantity of available data are crucial factors that directly impact the accuracy and reliability of these models. The scarcity and imbalance of datasets in certain domains pose challenges for researchers and practitioners, and the need for effective solutions is more pressing than ever. In this context, K-CGAN provides a promising approach to address data imbalance and improve the performance of machine learning models. Our results show that K-CGAN can be applied to different datasets with different characteristics, making it a valuable tool for data scientists and practitioners in various fields.

Acknowledgements

I am deeply grateful to my supervisor, Professor Simant Prakoonwit, for his unwavering support and guidance throughout my entire PhD journey. His expertise and faith in my abilities were instrumental in helping me successfully complete this challenging endeavor. Even during the toughest times, his motivational words and insightful yet critical suggestions pushed me to constantly strive for improvement.

I am also immensely thankful to Bournemouth University, a place that has not only provided me with a world-class education but has also introduced me to a community of inspiring individuals. The collaborative and nurturing environment fostered at the university has played a significant role in shaping my academic and personal growth.

Furthermore, I cannot express enough gratitude towards my family, especially my mother, for her unwavering support and understanding during the difficult phases of my life. Although my father is no longer with us, his legacy of strength and determination continues to inspire me every day. I owe them a tremendous debt of gratitude for their unwavering dedication, love, and constant motivation that kept me going through the challenges of my PhD journey. I consider myself incredibly proud and privileged to have such supportive parents who have been my pillars of strength.

I am deeply honoured and grateful to have received several prestigious Awards at the IEEE 2022 International conferences. These accolades, including the Significant Contribution Award at CAIT 2022, the Best Presentation Award at the same conference, the Best Paper Award at CoNTESA '22, and the Excellent Presentation Award at ICPES 2022, represent a culmination of dedicated effort and passion in the field of computer science and artificial intelligence. I extend my heartfelt gratitude to the organizers, reviewers, and fellow participants for their recognition and support. These Awards serve as a motivating force to continue pushing boundaries and contributing meaningfully to the advancement of these critical domains. Thank you for this tremendous honour.

In conclusion, I am truly humbled and honoured to have had the privilege of working with remarkable mentors, being part of a nurturing academic community, and having the unwavering support of my family. Their collective contributions have shaped me into the researcher I am today, and for that, I will forever be grateful.

Contents

Abstract	3
List of Publications	13
Chapter 1	15
1.0 Introduction	15
1.1 Background of the Problem	16
1.1.1 Statement of the Problem.....	17
1.1.2 Purpose of the Study	17
1.1.3 Objectives	18
1.1.4 Impact of the Study.....	18
1.1.5 Importance of the Study.....	18
1.2 Link to Existing Knowledge.....	19
1.2.1 Industry Insights	19
1.3 Contributions of This Thesis	20
1.4 Thesis structure.....	22
Chapter 2	23
Literature review	23
2.1 Introduction	23
2.2 Algorithm-Level Approaches	25
2.3 Data-Level Approaches.....	27
2.3.1. Sampling Based Techniques.....	33
2.3.2. GAN-Based Techniques	34
2.4 Classifiers	38
2.4.1 XGBoost	38
2.4.2 Random Forest.....	38
2.4.4 MLP	39
2.4.5 Logistic Regression.....	39
2.5 Overview of State-of-the-art Methods	39
2.5.1 Resampling (Oversampling and Undersampling).....	40
2.5.2 SMOTE.....	41
2.5.3 ADASYN.....	43
2.5.4 Ensembling methods.....	43
2.5.4 GAN based methods	44
2.5.5 Sampling vs GAN-based techniques	48
2.5.6 Summary.....	48
2.6 Methodology.....	49
Chapter 3	51
Kullback-Leibler Divergence Conditional GAN (K-CGAN)	51
3.1 Introduction	51
3.2 K-CGAN Architecture and Implementation.....	51
3.2.1 KL-Divergence Loss Function.....	62
3.2.2 The Binary Cross-Entropy Loss.....	64
3.2.3 Loss Functions of K-CGAN	65
3.2.4 Advantages of using Novelty Loss	70

3.2.5 Disadvantages of using Novelty Loss	70
3.3 The Discriminator and the Generator Architectures of K-CGAN	71
Chapter 4	77
Novelty Loss development: Implementation and Experiments using Multiple Methods.....	77
4.1. Introduction	77
4.2 Datasets Pre-processing and Architectures.....	77
4.3 Experimental Settings	78
4.3.1 Hyperparameter Settings of GAN-based Oversampling Methods.....	79
4.3.2 Hyperparameter Settings of Oversampling Methods.....	81
4.3.3. Hyperparameter Settings of Classification Methods	82
4.3.4 Original and Balanced Datasets using Oversampling Techniques	83
4.3.5 Generator and Discriminator Architectures of GAN based Methods	84
4.3.6 GAN Training: Generator and Discriminator Losses	88
4.3.7 Results and Comparisons of Classification Models.....	90
4.4 Development of K-CGAN Framework: Impact of Custom Loss	97
4.4.1 Experiment 1: K-CGAN with Novelty KL Loss without SMOTE.....	98
4.4.2 Experiment 2: K-CGAN with Novelty KL Loss with SMOTE.....	99
4.4.3 Experiment 3: K-CGAN without KL Loss with SMOTE.....	102
4.4.4 Experiment 4: K-CGAN with Novelty KL Loss without SMOTE.....	104
4.4.5 Comparison and Analysis of 1-4 Experiments	105
4.4.6 PCA Representation Analysis.....	109
4.4.7 Impact of KL Loss in Training	111
4.4.8 Experiments with Batch Normalisation	112
4.5 GAN-based methods Hyperparameter tuning with credit card fraud and breast cancer data.....	113
4.5.1 Hyperparameter tuning with credit card fraud data	113
4.5.2 Hyperparameter tuning with breast cancer data.....	121
4.6 Optimised K-CGAN Novelty Loss Evaluation comparison with other methods on credit card fraud data.....	129
4.6.1 Hyperparameter Settings.....	131
4.6.2 Results Analysis.....	136
4.6.3 Classification performance with original dataset.....	148
4.6.4 Classification performance with balanced dataset using Novelty K-CGAN	149
4.6.5 Classification performance with balanced dataset multiple models comparison.....	150
4.6.6 Impact of Oversampling using Novelty K-CGAN	155
4.7 Optimized Novelty Loss Evaluation comparison breast cancer data	156
4.7.1 Hyperparameter Settings.....	157
4.7.2 Results Analysis.....	163
4.7.3 Classification performance with original breast cancer data	173
4.7.4 Classification performance with balanced dataset	174
4.7.5 Classification performance with balanced dataset multiple models comparison.....	176
Chapter 5	181
Discussion	181
Chapter 6	189
Conclusion and Future Work	189
6.1 Contributions	190
6.2 Further work.....	192

List of References.....193

Appendix.....209

List of Figures

Figure 1: GANs Architecture.....	35
Figure 2: Techniques to handle imbalanced classification issues.....	40
Figure 3: Experiment flow.....	50
Figure 4: Process steps of CGAN architecture	53
Figure 5: Architecture of K-CGAN method.....	54
Figure 6: For similar probability distributions KL Divergence is closer to 0 and for dissimilar probability distributions they are high values highlighting divergence	55
Figure 7: TensorFlow’s representation of K-CGAN: (a) Discriminator; (b) Generator and (c) Network Layers & Training Architecture.....	72
Figure 8: The WGAN Generator (a) and Discriminator (b), SDG GAN Generator (a) and Discriminator (b), NS GAN Generator (e) and Discriminator (f), LS GAN Generator (g) and Discriminator (h), K-CGAN Generator (j) and Discriminator (k)	85
Figure 9: (a) The discriminator and generator losses of WGAN, (b) the discriminator and generator losses of SDG GAN (b), (c) the discriminator and generator losses of LS GAN, (d) the discriminator and generator losses of NS GAN, (e) the discriminator and generator losses of K-CGAN	88
Figure 10: (a) ROC curve for original imbalanced dataset, (b) ROC curve for balanced dataset utilising SMOTE, (c) ROC curve for balanced dataset utilising ADASYN, (d) ROC curve for balanced dataset utilising B-SMOTE, (e) ROC curve for balanced dataset utilising WGAN, (f) ROC curve for balanced dataset utilising SDG GAN, (g) ROC curve for balanced dataset utilising NS GAN, (h) ROC curve for balanced dataset utilising LS GAN, (i) ROC curve for balanced dataset utilising K-CGAN.....	90
Figure 11: Cosine Similarity strategy	97
Figure 12: Distribution charts showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 for experiment 1.....	99
Figure 13: Distribution charts showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 for experiment 2 with implementation of SMOTE to assist with K-CGAN (with KL divergence) training	101
Figure 14: Distribution charts showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 experiment 3 without KL Loss GAN with SMOTE	103
Figure 15: Distribution chart showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 experiment 4...105	105
Figure 16: PCA comparison original and generated data: (a) and (b) experiment 1, (c) and (d) experiment 2, (e) and (f) experiment 3, (g) and (h) experiment 4	110
Figure 17: K-CGAN discriminator loss without batch normalisation (a), K-CGAN discriminator loss with batch normalisation (b), K-CGAN discriminator loss extremely large networks (c)	112
Figure 18: K-CGAN generator (a) and discriminator (b) loss credit card fraud data	130
Figure 19: Correlation metric comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN credit card fraud data.....	136
Figure 20: Single column ‘Amount’ distribution comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN credit card fraud data (a), ‘V1’ (b), ‘V2’ (c), ‘V3’(d), ‘V4’ (e), ‘V5’ (f),‘V6’ (g), ‘V7’ (h), ‘V8’ (i), ‘V9’ (j), ‘V10’ (k),‘V11’ (l), ‘V12’ (m), ‘V13’ (n), ‘V14’ (o), ‘V15’ (p), ‘V16’ (q), ‘V17’ (r), ‘V18’ (s), ‘V19’ (t), ‘V20’ (u), ‘V21’ (v), ‘V22’ (w), ‘V23’ (x), ‘V24’ (y), ‘V25’ (z), ‘V26’ (z2), ‘V27’ (z3), ‘V28’ (z4).....	140
Figure 21: Bi-variate distribution ‘Amount vs V6’ comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with Credit card fraud data	

(a), ‘Amount vs V7’ (b), ‘Amount vs V14’ (c), ‘V1 vs V3’ (d), ‘V1 vs V4’ (e), V1 vs V5’ (f), ‘V1 vs V7’ (g), V1 vs V10 (h), ‘V1 vs V11’ (i), ‘V1 vs V16’ (j), ‘V1 vs V17’ (k), ‘V1 vs V5’ (l), ‘V2 vs V11’ (m), ‘V3 vs V5’ (n), ‘V3 vs V7’ (o).....	146
Figure 22: ROC curves (a) original imbalanced dataset, (b) balanced dataset with SMOTE, (c) balanced dataset B-SMOTE, (d) balanced dataset with ADASYN, (e) balanced dataset with Vanilla CGAN, (f) balanced dataset with WGAN, (g) balanced dataset with SDG GAN, (h) balanced dataset with NS GAN, (i) balanced dataset with LS GAN, (j) balanced dataset with K-CGAN	151
Figure 23: K-CGAN generator (a) and discriminator (b) loss breast cancer data	156
Figure 24: Correlation metric comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN breast cancer data	163
Figure 25: Single column ‘area_mean’ distribution comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN breast cancer data (a), ‘area_se’ (b), ‘area_worst’ (c), ‘compactness_mean’ (d), ‘compactness_se’ (e), ‘compactness_worst’ (f), ‘concave points_mean’ (g), ‘concave points_se’ (h), ‘concave points_worst’ (i), ‘concavity_mean’ (j), ‘concavity_se’ (k), ‘concavity_worst’ (l), ‘fractal_dimension_mean’ (m), ‘fractal_dimension_se’ (n), ‘fractal_dimension_worst’ (o), ‘perimeter_mean’ (p), ‘perimeter_se’ (q), ‘perimeter_worst’ (r), ‘radius_mean’ (s), ‘radius_se’ (t), ‘radius_worst’ (u), ‘smoothness_mean’ (v), ‘smoothness_se’ (w), ‘smoothness_worst’ (x), ‘symmetry_mean’ (y), ‘symmetry_se’ (z), ‘symmetry_worst’ (z1), ‘texture_mean’ (z2), ‘texture_se’ (z3), ‘texture_worst’ (z4)	167
Figure 26: Bi-variate distribution ‘area_mean vs symmetry_worst’ comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla GAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN on breast cancer data (a), ‘compactness_worst vs symmetry_worst’ (b), ‘concavity_se vs symmetry_se’ (c), ‘fractal_dimension_worst vs area_se’ (d), ‘fractal_dimension_worst vs radius_se’ (e), ‘fractal_dimension_worst vs symmetry_worst’ (f), ‘perimeter_se’ vs concavity_mean’ (g), ‘perimeter_worst vs radius_mean’ (h), ‘radius_mean vs smoothness_se’ (i), ‘radius_se vs area_worst’ (j), ‘smoothness_se vs concavity_se’ (k), ‘smoothness_se vs perimeter_se’ (l), ‘symmetry_se vs compactness_worst’ (m), ‘symmetry_worst vs compactness_se’ (n), ‘texture_mean vs fractal_dimension_se’ (o)	171
Figure 27: ROC curves (a) original imbalanced dataset, (b) balanced dataset with SMOTE, (c) balanced dataset ADASYN, (d) balanced dataset with B-SMOTE, (e) balanced dataset with Vanilla CGAN, (f) balanced dataset with SDG GAN, (g) balanced dataset with NS GAN, (h) balanced dataset with WGAN, (i) balanced dataset with LS GAN, (j) balanced dataset with K-CGAN	176

List of Tables

Table 1: Novelty K-CGAN Optimised hyperparameter settings for credit card fraud data	58
Table 2: Novelty K-CGAN Optimised hyperparameter settings for Breast cancer data	59
Table 3: Vanilla CGAN, WGAN and NS GAN hyperparameter settings	80
Table 4: LS GAN, SDG GAN and Custom K-CGAN initial experimental hyperparameter settings	81
Table 5: SMOTE, ADASYN and B-SMOTE hyperparameter settings.....	81
Table 6: Classification methods hyperparameter settings.....	82
Table 7: Original imbalanced credit card fraud dataset	83
Table 8: Balanced credit card fraud dataset by oversampling the minority class with SMOTE, B-SMOTE and ADASYN and GAN based methods.....	83
Table 9: Classifiers performance on original imbalanced dataset	91
Table 10: Classifiers performance on balanced dataset SMOTE oversampling the minority class.....	92
Table 11: Classifiers performance on balanced dataset ADASYN oversampling the minority class.....	92
Table 12: Classifiers performance on balanced dataset B-SMOTE oversampling the minority class.....	93
Table 13: Classifiers performance on balanced dataset using SDG GAN by oversampling the minority class.....	94
Table 14: Classifiers performance on imbalanced dataset NS GAN oversampling the minority class	94
Table 15: Classifiers performance on imbalanced dataset using LS GAN to oversampling the minority class.....	95
Table 16: Classifiers performance on imbalanced dataset using Novelty Loss K-CGAN by oversampling the minority class.....	96
Table 17: Cosine Similarity Scores comparison of each variable in experiments 1, 2, 3 & 4.....	106
Table 18: LS GAN hyperparameter tuning parameters	113
Table 19: LS GAN top 5 experiments	114
Table 20: NS GAN hyperparameter tuning parameters.....	115
Table 21: NS GAN top 5 experiments.....	115
Table 22: SDG GAN hyperparameter tuning parameters	117
Table 23: SDG GAN top 5 experiments	117
Table 24: WGAN hyperparameter tuning parameters	118
Table 25: WGAN top 5 experiments	119
Table 26: Novelty loss K-CGAN hyperparameter tuning parameters	120
Table 27: Novelty loss K-CGAN top 5 experiments	120
Table 28: Novelty loss K-CGAN hyperparameter tuning large set of parameters for breast cancer data	121
Table 29: Novelty loss K-CGAN top 5 experiments large set of parameters for breast cancer data.....	123
Table 30: Novelty loss K-CGAN hyperparameter tuning narrow set of parameters for breast cancer data.....	125
Table 31: Novelty loss K-CGAN top 5 experiments utilising narrow set of parameters.....	126
Table 32: Novelty loss K-CGAN hyperparameter tuning least set of parameters for breast cancer data.....	127
Table 33: Novelty loss K-CGAN top 5 experiments utilising least set of parameters.....	127
Table 34: Balanced credit card fraud dataset using optimised methods	130
Table 35: Oversampling methods hyperparameter settings	131
Table 36: Vanilla CGAN, WGAN and NS GAN optimised hyperparameter settings	131
Table 37: LS GAN and SDG GAN optimised hyperparameter settings.....	132
Table 38: Novelty optimised K-CGAN hyperparameter settings for credit card fraud data	133
Table 39: Classification methods hyperparameter configuration settings	135
Table 40: Classifiers performance on original imbalanced credit card dataset	148
Table 41: Classification performance with balanced credit card fraud dataset using Novelty K-CGAN oversampling minority class	149
Table 42: Precision values for classification methods multiple methods comparison.....	152
Table 43: Recall values for classification methods multiple methods comparison	153
Table 44: F1 Score values for classification methods multiple methods comparison	154
Table 45: Accuracy values for classification methods multiple methods comparison	155
Table 46: Balanced breast cancer dataset using optimised methods.....	157

Table 47: Oversampling methods hyperparameter settings	157
Table 48: Vanilla CGAN, WGAN and NS GAN optimised hyperparameter settings	158
Table 49: LS GAN and SDG GAN optimised hyperparameter settings.....	159
Table 50: Novelty K-CGAN optimised hyperparameter settings for breast cancer data.....	160
Table 51: Classification methods hyperparameter configuration settings	162
Table 52: Classification methods on original imbalanced breast cancer data	173
Table 53: Classification performance with balanced breast cancer dataset using Novelty K-CGAN oversampling minority class	174
Table 54: F1 Score values for classification methods multiple methods comparison	177
Table 55: Accuracy Score values for classification methods multiple methods comparison	178
Table 56: Recall Score values for classification methods multiple methods comparison	179
Table 57: Precision Score values for classification methods multiple methods comparison.....	180
Table 58: Summary of Classifiers performance on original imbalanced credit card fraud data.....	184
Table 59: Classification performance with balanced credit card fraud data by leveraging K-CGAN for oversampling the minority class	184
Table 60: F1 Score values for classification methods multiple methods comparison credit card fraud data ...	185
Table 61: Summary of Classifiers performance on original imbalanced breast cancer data	186
Table 62: Classification performance with balanced breast cancer data using K-CGAN to oversample minority class.....	187
Table 63: F1 Score values for classification methods multiple methods comparison breast cancer data.....	187

Abbreviations

Adaptive Synthetic Sampling Approach (ADASYN)
Area under the ROC Curve (AUC)
Artificial Intelligence (AI)
Artificial Neural Network (ANN)
Borderline Synthetic Minority Oversampling (B-SMOTE)
Central Processing Unit (CPU)
Conditional GAN (CGAN)
Extreme Gradient Boosting (XGBoost)
False Negative (FN)
False Positive (FP)
Generative Adversarial Network (GAN)
Graphics Processing Unit (GPU)
Input/ Output (IO)
Kullback-Leibler Divergence Conditional Generative Adversarial Network (K-CGAN)
K-Nearest Neighbours (KNN)
Logistic Regression (LR)
Least Squares GAN (LS GAN)
Machine Learning (ML)
Multi-layer Perceptron (MLP)
Neural Network (NN)
Non-Saturating GAN (NS GAN)
Principal Component Analysis (PCA)
Random Forest (RF)
Receiver Operating Characteristic Curve (ROC)
Synthetic Data Generation GAN (SDG GAN)
Support Vector Machine (SVM)
Synthetic Minority Oversampling (SMOTE)
True Negative (TN)
True Positive (TP)
Wasserstein GAN (WGAN)

List of Publications

Journal Papers

Strelcena, Emilija and Prakoonwit, Simant, *Improving Cancer Detection Classification Performance Using GANs in Breast Cancer Data*. IEEE Access, 2023.

Strelcena, Emilija and Prakoonwit, Simant, *Improving Classification Performance in Credit Card Fraud Detection by Using New Data Augmentation*. AI Systems: Theory and Applications. International Journal on International Journal of Artificial Intelligence (AI), 2023.

Strelcena, Emilija and Prakoonwit, Simant, *Effective Feature Engineering and Classification of Breast Cancer Diagnosis, a Comparative Study*. Feature Paper in Computational Biology and Medicine. International Journal of BioMedInformatics, 2023.

Strelcena, Emilija and Prakoonwit, Simant, *A Survey on GAN techniques for Data Augmentation to address the Imbalanced Data issues in Credit Card Fraud Detection*. Privacy and Security in Machine Learning. International Journal of Artificial Intelligence (AI), 2023.

Conference Papers

Strelcena, Emilija and Prakoonwit, Simant, *Generating synthetic data for credit card fraud detection using GANs*. IEEE 2022 The Third International Conference on Artificial Intelligence Technology (CAIT 2022). Proceedings. Zhejiang, China.

Strelcena, Emilija and Prakoonwit, Simant, *GAN-based Data Augmentation for Credit Card Fraud Detection*. IEEE 2022 International Conference on Big Data (IEEE BigData2022). Proceedings. Osaka, Japan.

Strelcena, Emilija and Prakoonwit, Simant, *Comparative Analysis of Machine Learning Algorithms using GANs through Credit Card Fraud Detection*. IEEE 2022 3rd International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA '22). Proceedings. North Macedonia.

Strelcena, Emilija and Prakoonwit, Simant, *A New GAN-based data augmentation method for Handling Class Imbalance in Credit Card Fraud detection*. IEEE 2023 10th International Conference on Signal Processing and Integrated Networks (SPIN 2023) Proceedings. Delhi-NCR, India.

Awards

Significant contribution Award at IEEE 2022 3rd International Conference on Computers and Artificial Intelligence (CAIT 2022).

Best presentation Award of the conference at IEEE 2022 3rd International Conference on Computers and Artificial Intelligence (CAIT 2022).

Best Paper Award at IEEE 2022 3rd International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA '22).

Excellent presentation Award at IEEE 2022 12th International Conference on Power and Energy Systems (ICPES 2022).

Chapter 1

1.0 Introduction

Data imbalance is a common issue in many types of businesses and industries (Peng et al., 2022; Saripuddin et al., 2022) and the financial or health industries are no exception. Imbalanced data refers to datasets where one class is overrepresented and the other is underrepresented (He and Garcia, 2009). This can lead to Machine Learning (ML) models that are biased and inaccurate, as the algorithm may be inclined to favour the majority class and ignore key patterns in the minority class. As technology advances, new enterprise techniques have emerged, with the credit card network being one of them. Every industry—from the household appliance industry to the automobile, health or banking sectors, and everywhere in between—is susceptible to data imbalance. In finance, when a third party uses another user's credit card to make a transaction, card fraud has taken place. This can happen when the third party obtains the card, the cardholder's PIN, login, or other details. Electronic fraud includes, for example, the unlawful usage of personal credit card information (Asha and KR, 2021). Despite the introduction of fraud prevention systems, data imbalance remains a significant problem. The prevalence of fraudulent transactions has grown rapidly due to the expansion of e-commerce and payment services (Chandrakanth, 2023). Credible Reports claim that between 2000 and 2015, the amount of debit card and credit card fraudulent transactions rose rapidly (Fanai and Abbasimehr, 2023). Furthermore, it has been demonstrated that fraudulent credit card uses and illegal transactions represent 75–80% of financial value, although only comprising 10–15% of all fraud instances (Saia and Carta, 2019). As a result of this data imbalance, credit card users and the industry as a whole are suffering considerable losses. This has prompted the need for better solutions to mitigate the issue of data imbalance and reduce the losses caused by fraud.

Data imbalance is also a problem in breast cancer diagnosis (Trister et al., 2017). This is due to the fact that mammograms, the most common form of diagnosis, are often prone to False Positives (FP) and under-diagnosis, resulting in incorrect conclusions and delayed treatments (Shams et al., 2018). Moreover, despite recent advances in deep learning and computer vision, there is still a lack of resources and human error that can lead to inaccurate diagnoses. This is compounded by the fact that mammography often requires multiple techniques to improve its accuracy, such as noting two views per breast, double reading, analysis of previous mammograms, and yearly interval screening, which can be costly and time-consuming (Kowal et al., 2013; Ramik R., 2020). Clearly, data imbalance is a serious issue in the diagnosis of breast cancer and must be addressed to prevent costly losses and ensure the best outcomes for patients. In order to seamlessly evaluate and detect minority activity given the volume of majority class samples numerous companies and firms significantly depend on Artificial Intelligence (AI). Moreover, machine learning-based techniques for detecting fraud with credit cards have grown into practice. Classification imbalances (Xie et al., 2019), features redundant complexity (Singh et al., 2021), validating latency (Zheng et al., 2018), data characteristic pre-processing, and concept drift were the key elements in the network training stage of the overall detecting fraud process (Ni et al., 2023). Further, classification technique that can discriminate between legitimate and illegal payments was a common method for addressing the issue of detecting fraud (Fanai and Abbasimehr, 2023). Building reliable and precise fraud detection systems requires incorporating the raw data from the fraud datasets into a lesser form. Investigators can assess whether entering a transaction was fake by using deep learning (Alejo et al., 2013). Further, similar to this, (Sanober et al., 2021) contended that resampling the data is a successful method for changing the distribution of imbalanced datasets. To improve the classifier's further development this may be done. That is only feasible, though, if they eliminate noisy information, minimise the extent of imbalance, watch out for data lost, and maintain sample points that are valuable for the classifier's training. Nevertheless, the reliability of the training datasets has a positive influence on how well machine learning algorithms function (Sanober et al., 2021), and the imbalance in the data is an ongoing challenge. The data typically shows a tiny portion of minority cases. This has a substantial impact on how well a machine-learning system can identify minority class samples.

Because classifier systems for AI are designed for well-proportioned training datasets, skewed data provides a special challenge. According to Xue and Zhang (2016), classifying all data as classifications with overwhelming samples may improve the accuracy of classification. Furthermore, according to Bahnsen et al. (2016), there are several difficulties with the task of identifying minority activity employing classification approaches. Such as class imbalance (Salekshahrezaee et al., 2023), cost sensitivity (the outlays of incorrectly labelling deceitful and ordinary transaction records are not the same) (Chandranth, 2023), temporal dependence among transactions (Karthika and Senthilselvi, 2023), concept drift (Van Belle et al., 2023), and necessitating classifier updates (Ahmad et al., 2023). Additionally, enthused by the efficacious claim of the deep learning approaches in numerous areas for instance computing vision (Dev et al., 2021; Sultana et al., 2020; Xue and Qin, 2022), translation (Luong et al., 2015), speech recognition (Chorowski et al., 2015), and forecasting complex time series data (Abbasimehr and Paki, 2022; Abbasimehr et al., 2020).

Further, there are different machine learning algorithms used to resolve the problem of class imbalance in credit card transactions. Machine learning (Abdulhayan et al., 2023), deep learning (Alharbi et al., 2022), data mining (Esmail et al., 2023), genetic programming (Prusti et al., 2023), and fuzzy logic (Kumar and Gupta, 2023) are all examples of cutting-edge technologies that have led to novel approaches of identifying various types of credit card fraud (Ashwin et al., 2023). There is still room for improvement in the efficiency of current AI-based algorithms for classifying minority classes due to the significant duplication of feature data and imbalance of class distribution in transactions (Ni et al., 2023). As a result, optimised feature engineering and sampling strategies must be incorporated into fraud detection algorithms.

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are a powerful tool for augmenting high-dimensional data (Ullah and Mahmoud, 2021). However, there is limited research on their use in fraud detection or cancer diagnosis. This research seeks to address this gap by obtaining a large number of valid data points that are representative of the minority class, such as fraudulent transactions or malignant tumours. Having more data like this can be used to train a binary classifier so it's effective against minority class, for instance, fraud or cancer diagnosis. Further, Numerous GAN-based algorithms, including Vanilla GAN (Goodfellow et al., 2014), Non-Saturating GAN (NS GAN) (Shannon et al., 2020), Conditional GAN (CGAN) (Mirza & Osindero, 2014), Synthetic Data Generation GAN (SDG GAN) (Charitou et al., 2021), Least Squares GAN (LS GAN) (Mao et al., 2017), and Wasserstein GAN (WGAN) (Arjovsky et al., 2017) as well as sampling based techniques Adaptive Synthetic Sampling Approach (ADASYN) (He et al., 2008), Synthetic Minority Oversampling (SMOTE) (Chawla et al., 2002) and Borderline Synthetic Minority Oversampling (B-SMOTE) (Han et al., 2005). However, these techniques have some problems and limitations. In our research we're exploring the potential of GANs to address the issue of data imbalance, as well as introducing a novel GAN technique called K-CGAN to address the limitations of the existing GAN models.

1.1 Background of the Problem

This section provides a succinct overview of the current technologies in research before discussing and highlighting the knowledge gaps that should be addressed.

Data scarcity is a significant problem since a large quantity of data is required in minority class detection to train deep learning models. Data augmentation is one of the most efficient ways to deal with this problem (Bansal et al. 2021; Langevin et al. 2021). In recent years, researchers have conducted various studies in this particular area. One of the main issues being faced while training datasets is that there is limited training data available in many application domains (Bansal et al. 2021). In some areas, data collection is not possible. For instance, training data is not possible in credit card fraud or breast cancer detection due to privacy concerns. Data collection is a time-consuming and costly task. On the contrary, financial and medical institutions need extensive data to detect minority class cases. An effective way to deal with limited training data issues is data augmentation. It is a method used to generate data from the existing data synthetically. Data augmentation saves both time and cost in gathering required data. Furthermore, it decreases the issue of sample inadequacy in deep learning models (Langevin et al. 2021, Shorten and Khoshgoftaar 2019). Besides limited training data, lack of relevant data is also a fundamental challenge to regress datasets. Large quantities of relevant data are required to improve the accuracy of deep learning models. Data augmentation can provide solutions through different methods to enhance the size and quality of training datasets to gain a better outcome (Bansal et al. 2021; Langevin et al.

2021). Furthermore, model overfitting is also regarded as a big challenge. Deep learning models require significant data to avoid the issue of overfitting. Overfitting is a modelling error when a model too closely fits the available dataset. In addition, when a model is trained on an inadequate dataset, it will be difficult for the model to generalise it perfectly for a new dataset. In addition, when these models are tested for any new data, they will not provide accurate predictions, making the model impractical. Therefore, the model needs more datasets to deal with the challenge of overfitting (Bansal et al. 2021). However, data augmentation lessens the issue of overfitting by training the model with a large set of appropriate data. Furthermore, data augmentation regularises the model and enhances its ability of generalisation.

Besides the above challenges, imbalanced data is also a significant problem to deal with real-life applications (Salekshahrezaee et al., 2023). This problem is prevalent in financial and medical institutions, especially with credit card fraud detection, as fraud transactions are too few compared with legal transactions. In addition, deep learning models need a large quantity of data to classify correctly, but occasionally available data is imbalanced, which creates difficulty in training deep models and affects the overall accuracy. Data can be re-sampled to solve this challenge; however, data augmentation can help this issue by dealing with highly imbalanced datasets by creating data for training machine learning models (Bansal et al., 2021; Langevin et al., 2021). Augmentation of data can be done through several approaches, for instance, the adversarial approach developed by (Goodfellow et al., 2014), the heuristic approach by (Ratner et al., 2017), and the style transfer approach proposed by (Gatys et al., 2016), and so on.

GANs developed by Goodfellow et al., (2014), are used to augment data effectively. GAN is a class of generative models that can create new data based on actual training data. It comprises two Neural Networks (NN), the Generator and the Discriminator, which work in opposite directions. The first one creates new data instances from actual data, while the other evaluates the synthetic data for authenticity. The applicability of GANs is high, and they can be used in various fields, such as credit card fraud or breast cancer detection. Deep learning models have transformed our daily life since they are doing well in mitigating real-world challenges. However, the issue of data scarcity is a significant problem as a large quantity of data is required to test the authenticity of data. On the other hand, augmentation of data is an effective way to deal with scarce data. The issue with data imbalance is also a major challenge and deep learning models, once sufficiently trained, can be effective in balancing the dataset and improving classification performance.

GANs have revolutionised the AI field, but there is still much work to be done in order to apply them effectively within the financial and medical sectors, as well as address their limitations. GANs are very successful in generating synthetic data, however, due to their current limitations such as mode collapse, unstable training, and difficulty of convergence, to name a few, GANs can be a challenge to implement. More research is needed to resolve these challenges.

1.1.1 Statement of the Problem

Due to the highly imbalanced nature of many datasets in machine learning, existing models for supervised learning are often unable to accurately identify rare events. For example, credit card fraud datasets typically contain a majority of valid transactions and a small minority of fraudulent ones. Similarly, medical imaging applications such as breast cancer detection usually have far more examples of healthy tissue than cancerous ones. This problem is compounded by the fact that imbalanced datasets can cause supervised learning algorithms to suffer from poor performance or even bias. As a result, there is an urgent need for better models that are able to address the challenge of imbalance in order to improve accuracy and reliability.

The potential for GANs to address this challenge has been demonstrated in existing research studies (Goodfellow et al., 2014; Shannon et al., 2020; Mirza & Osindero, 2014; Charitou et al., 2021; Mao et al., 2017; Arjovsky et al., 2017). However, further development of GAN based models is needed in order to provide an effective solution that can be applied across a wide range of datasets. In particular, optimised GAN architectures are required in order to ensure a high degree of accuracy and reliability when dealing with imbalanced datasets.

1.1.2 Purpose of the Study

This research aims to develop a robust machine learning methodology capable of generating high quality synthetic dataset, resolving class imbalance and thus improving classification algorithms accuracy in detecting minority class. By developing a new GAN approach for re-balancing data, providing an optimised system to

guide the training process and validating its performance on widely adopted datasets, it is hoped that this research will provide insight into how these machine learning technologies can effectively be deployed in real-world applications.

1.1.3 Objectives

Specifically this research may be divided into four main objectives:

- 1.** To develop a new GAN technique to generate high quality re-balanced data. Thus, by balancing the dataset improving classification performance of defecting minority classes. The research focuses on developing an effective model to generate synthetic dataset, rebalance dataset, and improve classification performance in detecting fraud in credit card transactions as well as breast cancer detection. In doing so, a new approach is proposed whereby data has been re-balanced and custom hyperparameter optimised GAN architecture along with the Kullback-Leibler divergence loss has been applied in order to reduce the noise and increase accuracy.
- 2.** To develop optimised hyperparameter fine-tuned training processes of the GAN method (once the GAN is trained, human intervention is not required).
- 3.** To validate the primary performance metrics and evaluate the accuracy of the proposed GAN on widely adopted datasets such as real-world credit card fraud and breast cancer datasets. This will involve running experiments on these data sets and assessing its primary performance metrics such as f1-score, precision, recall and accuracy, and comprehensive data quality evaluation.
- 4.** The research also aims to evaluate how this system can be effectively deployed to cloud-based platforms. This will involve exploring the scalability of the proposed GAN and identifying any issues that may arise from its deployment. This will allow for an effective scaling up of the system to handle larger datasets as well as future proofing it against any potential bottlenecks.

1.1.4 Impact of the Study

The results of this study can also be used to inform the future development of more effective minority class detection systems. The proposed GAN method could prove to be an innovative and reliable approach for generating minority class transactions in real-time, which would have far reaching benefits for both individuals and organisations. By introducing a new model it will provide a cost-effective solution that can help generate synthetic data in the finance field. Additionally, the study aims to contribute to the field of medical breast cancer.

This research could also provide useful insights into how GANs can be used to generate other types of data and by resolving data imbalance help organisations improve classification accuracy. This would be an invaluable resource to the industry and could lead to further advancements in credit card fraud and breast cancer detection systems by developing effective synthetic data generation methods and resolving data imbalance challenges. The innovative GAN approach for generating minority class transactions holds the promise to transform data handling practices across diverse domains. As organizations increasingly depend on data-driven decision-making, addressing data imbalance becomes crucial to enhance the performance of machine learning algorithms.

1.1.5 Importance of the Study

The study to address data imbalance issues using GANs is important for many reasons. Firstly, it has the potential to improve current methods and in the context of real life scenarios facing imbalanced data such as improving detection of credit card fraudulent activity would help protect customers and businesses from being victims of such crimes. Further example of breast cancer detection where early cancer detection could save patients' lives. As GANs utilise machine learning algorithms to identify patterns in large datasets, they are better able to detect target activity than traditional methods. Furthermore, this type of detection is more easily scalable and can be implemented across a variety of platforms. On the other hand, organisations can reduce their costs associated with investigations and reimbursements that occur as a result of fraudulent activity. Furthermore, implementing GAN-based detection can be more cost-effective than other methods, as it requires less manual labour to set up and maintain. Finally, this research will be a significant contribution to the development of more effective solutions for imbalance data and help in credit card fraud and breast cancer detection domains. By testing the performance of a GAN model against these datasets, we are able to gain insights into the best practices

for implementing such detection methods. This research will also help to inform future developments in solutions to imbalance data and assist in credit card fraud or breast cancer detection. Overall, this study is significant as it has the potential to improve current methods of imbalance data and detecting target activity and benefit individuals and organisations alike.

1.2 Link to Existing Knowledge

Our new proposed K-CGAN method is a GAN based approach proposed in our research study to enhance the performance of classifiers generating high quality synthetic data. It's based on CGAN architecture (Mirza and Osindero, 2014) with custom architecture loss functions and custom optimised hyperparameters. K-CGAN was tested on credit card fraud and breast cancer datasets to demonstrate its effectiveness in data generation and resolving class imbalance issues.

1.2.1 Industry Insights

Our proposed K-CGAN framework has proven to be highly efficient in addressing the issue of data imbalance through its optimised architecture. This has translated into improved results, making it a valuable tool for various types of datasets. The capability of the framework to work with such datasets ensures that it provides a comprehensive solution to the industry's data imbalance challenge. This paves the way for the use of GAN based models in a variety of applications, from medical diagnostics to credit card fraud detection.

In particular, this method could be beneficial in tasks that require accurate and reliable classification results in an environment with scarce or imbalanced data. It is expected that further research and experimentation will lead to even better performance. Thus, GAN based models are becoming increasingly attractive to industry practitioners seeking a reliable and effective solution to data imbalance. Furthermore, the use of GANs in other domains such as natural language processing and image recognition is growing rapidly due to their robustness and scalability. As the technology continues to mature, it is anticipated that the applications of these models will become more widespread, providing greater accuracy and efficiency across a wide range of fields. With its potential to revolutionise data analysis and classification, GAN based models are sure to be an important tool for industry professionals in the near future. Ultimately, this could lead to more reliable and accurate data-driven decisions that can benefit both businesses and consumers.

1.3 Contributions of This Thesis

In recent years, imbalanced datasets have been a major challenge in various domains including credit card fraud and breast cancer classification. To address this challenge, the K-CGAN model has been introduced in this research, offering an effective solution to generate synthetic data. The model has been tested using classification techniques, and its performance has been compared with the baseline models. The results have shown that the K-CGAN model outperforms the baseline models, with a significant improvement observed in credit card fraud detection tasks. The K-CGAN model introduces a new architecture that includes a custom loss function and custom optimized hyperparameters. This architecture is capable of generating high-quality synthetic datasets, which can be used to augment existing datasets, resolve data imbalance issues thus providing better accuracy and performance in classification tasks. This is a crucial contribution to the field of deep learning methods and machine learning classification techniques, as the K-CGAN model can be applied to various fields and tasks. The main areas of contributions are outlined below:

- **Kullback-Leibler Divergence Conditional Generative Adversarial Network (K-CGAN):** The K-CGAN network is proposed to improve the quality of the generated synthetic minority data to improve the performance of classification by balancing the dataset. The results show that K-CGAN generates high quality synthetic datasets and improves the classification performance significantly when compared to other augmentation techniques. The K-CGAN novel method is published in the Journals IEEE Access (Strelcena and Prakoonwit, 2023), AI Systems: Theory and Applications (Strelcena and Prakoonwit, 2022), conference papers IEEE: International Conference on Big Data (Big Data) (Strelcena and Prakoonwit, 2022), International Conference on Computers and Artificial Intelligence Technologies (CAIT) (Strelcena and Prakoonwit, 2022), 10th International Conference on Signal Processing and Integrated Networks (SPIN) (Strelcena and Prakoonwit, 2023). Our research achieved four Awards at IEEE 2022 CAIT for significant contribution and best presentation, IEEE 2022 ICPES for excellent presentation and best paper award at IEEE 2022 Contesa international conferences.
- **Generative Oversampling data-driven approach to address the issue of imbalanced datasets on Credit Card Fraud Detection and Breast Cancer Diagnosis:** The method utilises the K-CGAN deep learning model, which incorporates a custom architecture and hyperparameters tailored to each dataset. This meticulous optimization process ensures superior performance and consistency across all classification methods, with a specific focus on credit card fraud and breast cancer detection. By leveraging this novel approach, we can achieve more accurate and reliable results in these critical domains.
- **Effective features engineering and evaluation of the existing data augmentation and classification methods:** Extensive feature engineering and evaluation of data augmentation and classification methods, our studies also published in journals BioMedInformatics (Strelcena and Prakoonwit, 2023), Make: Privacy and Security in Machine Learning (Strelcena and Prakoonwit, 2022) and conference papers IEEE: International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA) (Strelcena and Prakoonwit, 2022) achieving Best Paper Award at IEEE 2022 3rd International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA '22).
- **Custom Optimised GAN architecture:** The K-CGAN model incorporates a Conditional Generative Adversarial Network (CGAN) to generate synthetic data that not only mimics the original data distribution but also aligns with specific class attributes. It achieves this by conditioning the generator to produce fake data based on target class labels. The K-CGAN utilises different activation functions for the generator and discriminator networks. The generator employs the widely-used ReLU activation function, while the discriminator leverages the LeakyReLU activation function. This combination allows the model to effectively handle complex datasets and improve convergence. The generator

network utilises a combined loss function, consisting of the trained discriminator loss and the KL divergence, to ensure that generated samples closely match the original data distribution and improve the quality of synthetic data. Carefully optimised hyperparameters, including learning rate, dropout rate, neuron sizes, kernel initializer, and kernel regularizer, enhance the K-CGAN's performance, prevent overfitting, and ensure stable convergence. The activation function in the nodes is deliberately set to decrease noise's influence, making the K-CGAN robust to noisy data. Additionally, the K-CGAN addresses the exploding gradient problem by utilising the glorot_uniform kernel initializer, which can hinder the training process of deep neural networks.

- **Optimised hyperparameter settings were obtained for other GAN based methods:** Our research performed hyperparameter tuning to find the optimal hyperparameters for other GAN based methods, including Vanilla CGAN, NS GAN, LS GAN, SDG GAN and WGAN. The results of this study can greatly assist in making informed decisions when selecting the appropriate data augmentation method. By leveraging these findings, researchers and practitioners can enhance their understanding and effectively choose the most suitable approach for their specific needs.
- **Multiple classification and oversampling methods implementation:** Popular classification methods implemented and tested in different conditions with original imbalanced and balanced datasets. The results can be useful when selecting classification methods for minority class detection resolving important areas demonstrated in credit card fraud and breast cancer detection. We have demonstrated the impact of SMOTE's application to assist in GAN training. This contribution not only enhances the performance of GANs but also proves to be a valuable asset for various other methods in the field.
- **Extensive application of visualisation of synthetic and original data:** During our extensive experimentation, we have extensively applied visualisation techniques to both synthetic and original data. By leveraging these techniques, we were able to gain a deeper understanding of the underlying patterns and relationships within the datasets. To evaluate the quality of the datasets, we employed various methods, including cosine similarity approaches, bivariate and univariate correlations. These approaches not only provided quantitative metrics for assessing data quality but also revealed valuable insights into the characteristics and distribution of the data. Furthermore, our comprehensive analysis allowed us to thoroughly compare the performance of the K-CGAN method with other existing approaches, highlighting its superiority in terms of data quality and accuracy. With this in-depth exploration, we were able to uncover the true potential and value of our approach, paving the way for future advancements in the field.

Most importantly its ability to generate synthetic data, the K-CGAN model framework provides a system and process that is highly beneficial for resolving imbalanced dataset issues. These contributions have been discussed in detail in Chapter 6, highlighting the importance of the K-CGAN model in addressing the challenges posed by imbalanced datasets. The contributions made by this research are significant, and the K-CGAN model framework has the potential to enhance the accuracy and performance of classification tasks in various domains.

1.4 Thesis structure

This research is structured as follows.

Chapter 2 provides a comprehensive literature review, offering an in-depth overview of the technologies employed in this research. The review encompasses various areas, including GAN development, testing methodologies, classification performance evaluation techniques, data analysis approaches, and synthetic data generation methods to address dataset imbalance issues.

In Chapter 3, the K-CGAN framework is introduced, detailing its architecture, implementation steps, and providing optimized hyperparameter settings specifically tailored for credit card fraud and breast cancer datasets. The chapter highlights the significance of the K-CGAN framework in addressing the challenges faced in these domains and emphasizes the potential impact it can have as a reliable data augmentation method on improving classification performance.

Chapter 4 documents the detailed development process of the K-CGAN framework, starting from its initial stages to the final optimised custom version. It explores the impact of KL (Kullback-Leibler) loss and custom hyperparameter tuning on different datasets, presents comprehensive classification performance results, and compares the performance of other existing GAN and oversampling methods, such as SMOTE, B-SMOTE, and ADASYN. The chapter aims to provide a thorough understanding of the strengths and limitations of the K-CGAN framework in comparison to alternative approaches.

The objective of this thesis extends beyond knowledge acquisition and performance improvement. It also aims to provide a solid foundation for future system design and implementation in the field. Chapters 5 delves into the detailed analysis of the experimental results, discussing the implications, insights, and potential areas of further research based on the conducted experiments. Ultimately, this research presents the K-CGAN model as a promising and versatile solution applicable in various domains, such as credit card fraud and breast cancer detection. This study significantly contributes to the understanding of both fields, offering valuable insights and a robust solution for future implementation and advancement in the respective areas.

Chapter 2

Literature review

2.1 Introduction

Most production datasets are susceptible to class imbalance. The issue of class imbalance is important to resolve in machine learning because it can significantly affect the performance of classifiers and lead to biased predictions (Zhou et al., 2020). When there is a significant difference in the number of samples between classes, classifiers tend to bias towards the majority class, resulting in poor performance for the minority class. Consequently, this can have serious consequences in real-world applications where the minority class may be of particular interest or importance. For instance, in medical diagnosis, a classifier that biases towards the majority healthy class may miss important diagnoses for rare diseases (Zhou et al., 2020). Furthermore, traditional evaluation metrics such as accuracy may not be appropriate for imbalanced datasets and can provide misleading results. This can lead to incorrect conclusions about classifier performance and potentially harmful decisions based on these conclusions (Zhou et al., 2020). Therefore, it is essential to recognize the problem of class imbalance and take appropriate steps to combat it when dealing with machine learning and real-world applications. Given its versatility, machine learning is quickly becoming the industry standard for different applications. In machine learning, algorithms focus on the ability to learn and improve autonomously through exposure to relevant data. Different algorithms and, in some cases, statistical models are used in machine learning to enable computers to perform tasks automatically by learning the characteristics of the data.

Machine learning methods have since played an important role in automated minority class detection. With the use of machine learning, researchers can determine whether an incoming transaction is in the minority class (Alejo et al., 2013). However, the performance of machine learning techniques greatly depends on the quality of the training data (Sanober et al., 2021; Xue and Zhang, 2016) and the imbalance in the data is not a trivial issue, especially when credit card frauds are considered. In general, only a small percentage of fraudulent transactions are presented in the data. This significantly affects how a trained machine learning algorithm can correctly detect fraud cases. Machine learning techniques are framed for well-balanced training data; thus imbalanced data pose a unique problem to classifier frameworks. According to Xue and Zhang (2016), we can attain greater classification accuracy through the classification of all samples as the classification with the majority of samples. Similarly, Sanober et al. (2021) argued that resampling of the data is an effective way to alter the distribution of datasets that are not balanced. This can be performed to get better subsequent progress of the classifier. However, it is only possible if we remove noise information, lessen the intensity of the imbalance degree, ensure to reduce information loss, and keep sample points which are helpful for the learning of the classifier. To address this issue, there exist many data augmentation techniques, such as SMOTE (Chawla et al., 2002), ADASYN (He et al., 2008), B-SMOTE (Han et al., 2005), CGAN (Mirza & Osindero, 2014), Vanilla GAN (Goodfellow et al., 2014), WGAN (Arjovsky et al., 2017), SDG GAN (Charitou et al., 2021), NS GAN (Shannon et al., 2020), and LS GAN (Mao et al., 2017) to balance the data. These methods are capable of synthetically generating additional data to balance the majority and minority class distribution. It is noted that the conventional classification techniques achieve higher accuracy over the positive class and poor accuracy over the negative class. Hence, the classification ability of the binary classifiers typically

decreases in imbalanced datasets with the high imbalance rate. Past details reveal that most of the classifiers would lose their efficiency when the imbalance rate hits (Xue and Zhang, 2016). The SMOTE technique was introduced to reduce the shortcomings faced by the random over sampling method. Similarly, GANs were presented in order to address the limitations of SMOTE. In addition, multiple GAN variants have been developed recently to improve the accuracy and address limitations of GANs. Further, different data augmentation methods have various characteristics suitable for different applications. Our study presents an investigation into how different data augmentation techniques affect the performance of classification algorithms on imbalanced data.

According to Xue and Zhang (2016), machine learning is used to train machines how to manage data better and more effectively. ML can be used to deduce details from the extracted data. In more recent years, due to the availability of data, the demand for ML has been very high. Furthermore, the study used data level and algorithm level approaches to resolve the issue of class imbalance in data. Our current study is mostly concerned with data level approaches instead of algorithm level approaches to address the issue of class imbalance in data sets. Further, data-level methods use data augmentation techniques to improve the class distribution in the data and produce more balanced data as compared to algorithm level approaches, which are more suitable for classifiers performance. Moreover, our study employs a new technique, known as K-CGAN. Some examples of well-established classification algorithms, i.e., Extreme Gradient Boosting (XGBoost) (Chen & Guestrin, 2016), Random Forest (RF) (Breiman, 2001), K-Nearest Neighbours (KNN) (Cover & Hart, 1967), Multi-layer Perceptron (MLP) (Rosenblatt, F., 1957), and Logistic Regression (LR) (DeMaris, A., 1995) are then used to evaluate the performance of the data augmentation techniques. Our suggested K-CGAN method is also a similar attempt to address the class imbalance issue and improve the overall efficiency of machine learning techniques.

The standard oversampling algorithm SMOTE's results can often be too noisy when the majority and minority classes are hard to distinguish, as well as not being flexible enough to handle high-dimensional data. For this very reason, modifications such as Borderline SMOTE and ADASYN have been developed in an effort to improve classification accuracy by enhancing the distinction between these two types of classes. Though oversampling techniques can help generate new samples that appear similar to the original data on its surface, in detail these replicates may differ from one another. This is especially true when it is hard to extract features in a regularised manner from the imbalanced dataset. SMOTE's method presents a certain degree of risk due to its lack of consideration for the majority class when aggregating minority regions. This danger is especially pronounced in cases with imbalanced classes as, oftentimes, the minority group is minuscule compared with the larger one and thus more likely to encounter crossover issues.

Building on the success of generative models, GANs (Langevin et al., 2022) have gained momentum in recent years as a reliable and versatile way to approximate real data distributions. These networks are highly adaptable and particularly easy to comprehend due to their general safety factor. Specific examples that leverage this technology can be created with relative ease. For studies and surveys that work with restricted budgeting, a precision technique grants specialists and analysts a sense of control over the process. This can be especially beneficial when concentrating on narrowly defined speculation since inspections can then be systematically curated to suit certain restrictions. Thus, precise techniques supply researchers with an invaluable degree of accuracy while keeping costs low. Despite its adaptable and general nature, with the careful fine-tuning of GANs it is possible to eliminate any potential drawbacks. Ultimately this could lead to the creation of an optimised architecture design which can be implemented for various machine learning applications.

2.2 Algorithm-Level Approaches

The primary focus of the algorithm-level study is on enhancing an established classification method so that it can handle imbalanced data, with the ultimate goal of boosting minority class classification performance while still maintaining a high bar for overall accuracy (Soh and Yusuf, 2019). The issue of imbalanced data categorization may be addressed with cost-sensitive learning, which is currently one of the most popular algorithm-level techniques (Ding et al., 2023). The cost-sensitive approach aids in improving the identification accuracy of positive samples by guiding the classifier to adjust the weight of incorrectly classified positive samples. For instance, Fu et al., (2022) suggested a Cost-Sensitive Support Vector Machine (CSSVM), a cost-sensitive model that takes its cues from both Support Vector Machine (SVM) and the asymmetric Linear Exponential (LINEX) loss function. By assigning a separate cost to each event, the model can perform instance-level sensitivity learning. To address the issue of imbalanced data categorization, the SVM classifier employs a cost-sensitive loss function to regulate the expense of misclassifying positive and negative samples. Two cost-sensitive KNN classifiers, Direct-CS-KNN and Distance-CS-KNN, were suggested by Zhang (2020) to reduce the negative impact of incorrect labels. The algorithmic level is both more intuitive and more productive than the data level (Ding et al., 2023). As a result, it excels in the categorization of data within a given domain. However, although it is possible to enhance algorithms, this is not always the best approach. It is clear from the concept of the cost-sensitive technique that providing the matching cost-sensitive matrix is crucial to the design of the algorithm. In a cost-sensitive matrix, the weight setting is often determined by domain specialists and is thus extremely domain-specific. As an added downside, budget-friendly learning methods developed for one area are famously hard to adapt for use in a different one.

In another study, Xue and Zhang (2016) proposed a Gradient Boosting Decision Tree (GBDT) to explore the impacts of factors on traffic accident indicators. The results showed that the GBDT can identify and prioritise the influential factors on traffic accident prediction. In addition, findings showed that this model outperforms all classical machine learning models featuring a ‘black-box’ in accuracy and prediction. The study conducted by (Chen et al., 2018) is a recent comprehensive survey of machine learning systems. In their study, the authors provided an overview of techniques introduced for the evaluation of machine learning explanations. Furthermore, they identified the traits of explainability after reviewing the explanations of explainability. Their findings demonstrated that the qualitative metrics for both example-based and model-based explanations are mainly used for the evaluation of interpretability. Furthermore, credit card fraud detection using auto encoder-based clustering based on auto-encoders was proposed by (Ngwenduna and Mbuyha, 2021). The system, which features three hidden layers and clusters data using k-means, was evaluated on a European dataset and was shown to perform favourably when compared with other systems. To handle the disparity dataset and avoid noise, Paasch (2008) suggested a misrepresentation location framework with a non-overlapped risk-based bagging ensemble algorithm. Bagging models eliminate noise and outliers from datasets. The sacking model is a goal achieved by a group of students working together to take calculated risks. Bag creation solves the problem of skewed data, and Naive Bayes eliminates the problem of transactional noise. Using a NBRE, they were able to reduce the cost of detecting fraud by 2–2.5 times while increasing the accuracy by 5–10 percentage points. The NRBE model was identified as the most suitable for fraud detection and the most suitable for a business dynamic method.

Further, Jiang et al. (2018) utilised an approach that blended Bayesian-based hyper parameter optimization with tuning by eye. They achieved this by utilising two distinct public datasets, one including fraudulent transactions and the other containing legitimate ones from the real world. Compared with other methods, their proposed approach performed better in terms of accuracy, precision, and f1 score.

Since the ratio of fraudulent to legitimate transactions is relatively high, Kumar and Iqbal (2019) developed an ensemble learning approach to avoid class imbalance in data. They found that compared with neural networks, random forest is superior at detecting fraud incidents. Large credit card transactions were also used as an experimental variable. Ensemble learning combines different machine learning techniques, such as random forest and neural networks. The findings of Lamba (2020) showed that credit card theft has been on the rise over the past few years. Several techniques use machine-learning algorithms to identify fraudulent

transactions and prevent them from being processed. In order to determine the precision of fraud detection, Makki et al., (2019) developed an application that makes use of machine learning techniques such as the k-nearest neighbour, decision tree, extreme learning machine, support vector machine, and multilayer perceptron. Using a combination of kNN, SVM, and DT, they made use of web-based protocols such as simple object access protocol and representational state transfer to transmit data effectively between many incompatible systems. The results of five different machine learning algorithms were evaluated using a metric that measured how well they predicted the results. Although SVM outperformed competing algorithms by a margin of 81.63%, the hybrid system they presented achieved an even greater accuracy of 82.58%.

In their study, Chen and Wasikowski (2008) introduced a hybrid machine learning technique to predict bus passenger flow. They named it Scaled Stacking Gradient Boosting Decision Trees (SS-GBDT). The findings of their study revealed that this novel method outperformed conventional machine learning models and did well in handling multicollinearity between influential factors. Further, using random forest methods, Prusti and Rath (2019) developed a model to tackle the issue of class imbalance in data sets. The supervised machine learning technique known as the random forest algorithm relies on a Decision Tree, with performance measured by means of a confusion matrix. Assuming a 90% accuracy, the suggested technique is quite promising. While Zheng et al., (2018) argued that credit card usage has been increasing day by day for online purchasing, the authors pointed out that online shopping has enhanced the number of credit card fraud cases as well. They emphasised the need to stop these cases. Furthermore, they introduced a novel technique that integrates Spark with a deep learning framework. They also implemented various methods to resolve class imbalance issues in datasets. These methods were SVM, RF, KNN and Decision Tree. The findings of the comparative study showed that 96% accuracy was obtained for the training and testing of data sets.

In a study, Singh et al., (2022) pointed out that the classification of imbalanced class datasets has gained much attention across many domains, including fraud detection. This is due to the negative impact of overlapping on the achievements of imbalanced class learning. The suggested method of this study was based on an augmented R-value, which aimed to pick features that obtained data with the least overlap degree, thus improving the classification performance. Moreover, their study presented three feature selection frameworks, RONS, ROS, and ROA, designed via sparse feature selection to lessen the overlapping and carry out binary classification. In addition, the findings of their study suggested that their presented frameworks that feature selection techniques manage the variation of a false discovery rate at the time of the main features for the modelling process. Finally, their empirical study used four credit card datasets to check the performance of their methods. The findings confirmed that their methods are superior to classical feature selection techniques.

To better understand the state of the art in Master Card fraud detection using machine learning algorithms, Sethia et al., (2018) conducted a comprehensive literature review of the methods currently in use. The field has been the subject of a great deal of study. They argue that a more robust system that can adapt to any circumstance is required. However, these techniques have some limitations. For instance, Langevin *et al.* (2022) argued that algorithmic approaches to improving classifiers for imbalanced data may not be effective if the minority class is too small or too noisy because if the minority class is too small, it may be difficult for the model to learn meaningful patterns from the available data. In this case, even with algorithmic approaches such as cost-sensitive learning or ensemble methods, the model may still struggle to accurately classify minority class samples. Similarly, if the minority class is too noisy (i.e., contains a high proportion of mislabelled or irrelevant samples), it may be difficult for the model to learn useful patterns. In this case, pre-processing techniques such as data cleaning or feature selection may be necessary before applying algorithmic approaches (Karthik et al., 2022). The authors also note that in some cases, pre-processing techniques such as sampling methods may not be effective if the minority class is too small or too noisy. Further, it is pointed out that algorithm level approaches involve adapting present classifier learning algorithms to bias the learning toward the minority class (Galar et al., 2011). However, the research noted that these methods may not be feasible in all cases because they require special knowledge of both the corresponding classifier and the application domain. In particular, algorithm level approaches may not be feasible when there is limited knowledge about the underlying data generating process or when there are complex interactions between features that make it difficult to identify which features are most important for predicting the minority class

(Rtayli, 2022). In these cases, it may be more effective to use other techniques like ensemble methods or data augmentation to report the class imbalance problem.

Overall, while algorithm level approaches can be effective for improving classifiers on imbalanced datasets, they may not always be feasible or practical depending on the specific characteristics of the dataset and problem being addressed. Zhou et al., (2020) argued that algorithm level approaches heavily rely on the quality and quantity of data used for training. In their research the authors used patent data to train their deep learning classifiers. However, patent data may not always be available or may not be representative of all emerging technologies. Further, algorithm level approaches may not capture all aspects of emerging technologies because emerging technologies are complex and multifaceted, and algorithm level approaches may not be able to capture all the nuances and subtleties associated with them (Zhou et al., 2020). For example, some emerging technologies may have social or ethical implications that cannot be captured by algorithm level approaches. Furthermore, algorithm level approaches may not be able to adapt to changes in emerging technologies over time but emerging technologies are constantly evolving and changing, and algorithm-level approaches may become outdated or ineffective as new technologies emerge. Finally, algorithm level approaches require significant expertise in machine learning and data analysis to develop and implement effectively. This can limit their accessibility to researchers who do not have expertise in these areas (Zhou et al., 2020).

Moreover, Charitou et al., (2020) pointed out that the algorithm level approach relies on the assumption that the minority class can be represented by a low-dimensional manifold. This may not always be true in practice, and in cases where the minority class is highly complex or has high variability, the generated synthetic samples may not accurately represent the true distribution of the minority class (Charitou et al., 2020). Further, the framework is able to generate synthetic samples of minority-class data, but it does not provide any guarantees about their quality or usefulness. In some cases, the generated samples may be unrealistic or irrelevant to the problem at hand, which could negatively impact classifier performance (Charitou et al., 2020). Furthermore, the framework is able to address imbalanced data issues without relying on traditional oversampling or undersampling techniques, it still requires labelled data for training. This means that if labelled data is scarce or expensive to obtain, this approach may not be feasible. Finally, while the proposed framework outperforms other classification methods in terms of F1 score on several datasets used in their study, it is unclear how well it would perform on other datasets with different characteristics (Charitou et al., 2020). Further research is needed to evaluate its generalizability and robustness across different domains and applications.

2.3 Data-Level Approaches

The goal of a data-level technique is to resample the data until the negative and positive samples are about equal before classifying the data. Since the data-level processing approach is not dependent on the classification model, it is frequently employed to address the issue of imbalanced data sets that need to be categorised. Oversampling, undersampling, and hybrid sampling are the primary categories into which the individual data-level approaches fall. The goal of oversampling is to correct this difference by increasing the size of positive samples while leaving negative samples alone. Ding et al., (2023) argued that oversampling of positive samples may be broken down into two distinct categories: local information-based and global information-based. SMOTE oversampling is the most common technique used in local information-based oversampling (Ding et al., 2023). By performing random linear interpolation between nearby samples, the SMOTE technique creates additional positive samples (Chawla et al., 2002). The SMOTE algorithm has inspired the development of many similar but more advanced approaches, including ADASYN, SMOTE-ENN, LORAS, and many more (Maldonado et al., 2022). Although samples close to the decision boundary are crucial for classification, a lot of research has been done on how to reliably produce them (Wei et al., 2020). The global oversampling technique generates new data by taking into account the variance, mean value, and probability distribution of positive samples, as opposed to generating new samples only based on local information (Abdi and Hashemi, 2015). The combined probability distribution of data characteristics and Gibbs sampling was introduced by Das et al., (2014) as a method for creating new minority samples.

Since GAN has a high degree of accuracy when it comes to fitting data, it is often used for synthesis and the resolution of imbalanced learning issues (Fan et al., 2022). Consider the sequence GAN-based credit default collection and analysis technique presented by Fan et al., (2022) for the production of discrete data. In order to create credit default swap transaction data that is diverse and useful, this technique incorporates a reinforced learning approach into the original GAN network (Ger and Klabjan, 2019). The reduction of negative samples (undersampling) helps even out the distribution of classes while preserving the integrity of the data (positive samples) (Zhang et al., 2021). Undersampling's fundamental principle is to eliminate samples that have negligible influence on the total data distribution, to maintain a balance between the positive and negative data (Vuttipittayamongkol and Elyan, 2020). Xie et al., (2021) introduced density and distance as measures of sample significance, built a sampling sequence based on this importance, and then chose the most representative negative samples from this series. Further, data preparation can be improved via hybrid sampling, which combines oversampling and undersampling algorithms (Nabulsi et al., 2021).

Class decomposition, as suggested by Elyan et al., (2021) is one approach to optimising classification accuracy when dealing with imbalanced data. In order to deal with the problem of binary imbalanced data classification, Yu et al., (2021) suggested a hybrid classifier ensemble architecture (HCE). Adaptive two-stage undersampling (ATUP) and metric-based data space transformation (MDST) are the core components of the methodology. For a well-rounded dataset, they employ MDST to locate the proper embedding space and ATUP to pick representative samples (Yang et al., 2021). Traditional oversampling approaches create fresh samples locally, leading to poor generalisation capacity and inability to deliver improved classification judgements, and are thus among the methods based on the data level that have the potential to improve accuracy. Using an undersampling technique typically involves eliminating relevant data, which might alter the original data's distribution. Because of this, standard GAN methods frequently experience mode collapse and fail to account for the sparseness of positive class data in the class overlap region (Ding et al., 2023). Further, researchers have also used data-level approaches to address the imbalanced class challenge. These data-level approaches used the sampling technique to deal with this challenge.

Furthermore, to justify a data-level approach to be a better technique different studies have their own point of views. For instance, Abdallah et al., (2016) argued that the data level approaches such as undersampling and oversampling can be used in fraud detection systems to address class imbalance issues. This can lead to the solution of an issue of biased classification results, where the classifier tends to forecast the majority class more often than the minority class. Undersampling and oversampling techniques have been widely used in fraud detection systems (Abdallah et al., 2016). For example, Phua et al., (2010) utilised undersampling and oversampling techniques to balance imbalanced datasets. However, both undersampling and oversampling have their limitations. Undersampling may lead to a loss of information from the majority class while oversampling may lead to overfitting and reduced generalisation performance (Ng et al., 2020). In conclusion, the data level approach is an important technique used in fraud detection systems to address class imbalance issues and improve classification performance by balancing datasets and reducing noise (Singh et al., 2022). Further, López et al., (2013) argued that data-level approaches aim to modify the real dataset in some way to balance the class distribution before applying a classification algorithm. One common technique is oversampling, where instances from the minority class are replicated or synthesised to increase their representation in the dataset. Another technique is undersampling, where instances from the majority class are detached to decrease their dominance (López et al., 2013). Further, the authors cited several studies that have used oversampling and undersampling techniques with success. For example, Chawla et al., (2002) used random oversampling and undersampling on several imbalanced datasets and found that both techniques improved classification performance compared to using the original dataset. Similarly, He and Garcia (2009) used the SMOTE on several datasets and found that it outperformed random oversampling and undersampling.

However, López et al., (2013) noted that data-level techniques have some limitations. Oversampling can lead to overfitting if synthetic instances are too similar to existing ones (Gupta and Sharma, 2022) while undersampling can discard useful information from the majority class. Additionally, both techniques can increase computational complexity and training time (López et al., 2013). In conclusion, data-level approaches offer a way to address imbalanced datasets by modifying the original data before classification. While they

have shown promise in improving performance on certain datasets, they also have limitations that should be considered when choosing an approach for a particular problem (López et al., 2013). Singh et al., (2022) discussed the use of data-level approaches to handle class imbalance. The authors evaluated the performance of diverse data-level techniques to avoid the issue of class imbalance and employed several performance measures. Data-level approaches involve manipulating the training data to address the class imbalance (Singh et al., 2022). The study compared several data-level algorithms, including oversampling, undersampling, and hybrid sampling techniques (Kaur and Gosain, 2018). They noted that using a combination of different algorithms can improve performance. Singh et al., (2022) cited several previous studies that have used data-level approaches for fraud recognition, including Batista et al., (2004) and Phua et al., (2004). They also reference other studies that have compared different sampling techniques for imbalanced datasets, such as He et al., (2008). Overall, the research provided a detailed discussion of data-level approaches for class imbalance in datasets. The authors used these data-level approaches to manipulate the training data and improve model performance for class imbalance in datasets. They evaluate the performance of different algorithms using metrics such as accuracy and F1 score. These measures are generally employed in machine learning and classification tasks to evaluate model performance (Singh et al., 2022). The authors report these performance measures for each data-level algorithm evaluated in their study. They found that hybrid sampling techniques outperformed other methods in terms of F1 score and accuracy. Specifically, they reported an accuracy range from 0.977 to 0.994, an F1 score range from 0.758 to 0.919, and an Area under the ROC Curve (AUC) - Receiver Operating Characteristic Curve (ROC) range from 0.971 to 0.996 for different hybrid sampling techniques.

Werner de Vargas et al., (2023) discussed various data-level approaches that have been implemented to address the issue of data imbalance in machine learning. According to the article, data-level approaches involve modifying the original dataset by either undersampling the majority class or oversampling the minority class. The article cited several studies that have used oversampling techniques, such as SMOTE and ADASYN, to address data imbalance. A study by Batista et al., (2004) found that combining undersampling with oversampling using SMOTE resulted in better classification performance than using either technique alone. The article also discussed hybrid approaches that combine oversampling and undersampling techniques and cost-sensitive learning and ensemble learning methods (Werner de Vargas et al., 2023). The article provided a comprehensive overview of various data-level approaches that have been implemented to address data imbalance in machine learning. The study systematically maps and analyses the existing literature on imbalanced data preprocessing techniques in machine learning. The authors identified 364 relevant studies and analysed them based on several criteria, including the type of imbalanced data problem, the type of preprocessing technique used, and the performance measures reported. In terms of data-level approaches, the study found that oversampling and undersampling techniques were the most commonly used methods to address imbalanced data (Werner de Vargas et al., 2023). Specifically, oversampling techniques such as SMOTE and undersampling techniques such as Tomek links were frequently used in the studies analysed. The study also found that hybrid approaches, which combine oversampling and undersampling techniques, were effective in improving classification performance on imbalanced datasets. For example, combining SMOTE with ENN was found to be effective in several studies (Werner de Vargas et al., 2023).

Moreover, Hordri et al., (2018) discussed two methods to resolve the issue of imbalanced data. One of these methods is the oversampling and the second is undersampling. A combination of both methods can also be used to achieve a balanced dataset. The research cited several studies that have used these resampling techniques to tackle the class imbalance issue. Specifically, the authors divide their dataset into training and testing sets and then apply under and over sampling methods to balance the distribution of fraudulent and non-fraudulent transactions in the training set. They use ratios of 30:70 and 50:50 for fraudulent to legal transactions, respectively (Hordri et al., 2018). The authors then train several classification models on these balanced training sets. They evaluate the performance of these models using different performance methods such as F1-measure, sensitivity, accuracy, specificity, and precision. The results showed that these techniques can improve classification performance for various classification models. Several performance measures were used to evaluate the effectiveness of resampling techniques in class imbalance data. In the article, the accuracy value is 0.90, the precision value is 0.70, and the F1-measure value is 0.77.

Further, Zhou et al., (2020) used a data-level approach to address the issue of limited training samples when employing deep learning to estimate evolving technologies. Specifically, they propose a GAN-based data augmentation method to generate synthetic samples that are similar in distribution to the original ones. The data-level approach involves manipulating the input data directly, rather than modifying the model architecture or training process (Mohindru et al., 2021). By generating synthetic samples that are similar in distribution to the original ones, this approach can increase the size and diversity of the training dataset, which can improve the performance of deep learning models by reducing overfitting and improving generalisation (Tiwari et al., 2021). Overall, this study demonstrates how data-level approaches such as GAN-based data augmentation can be used to address the issue of limited training samples in deep learning applications (Zhou et al., 2020). Furthermore, the authors used several performance measures to appraise the efficiency of their suggested approach. The authors reported that the estimating accuracy extended 0.77 when using 1000 synthetic samples generated by their GAN-based data augmentation method. This means that out of all the samples in their dataset, 0.77 were correctly classified by their DNN classifiers. The precision, recall, and F1-score were also reported for each class in their dataset (Zhou et al., 2020). For the emerging technology (ET) class, the precision was 0.78, recall was 0.76, and F1-score was 0.77. For non-emerging technology (NET) class, the precision was 0.76, recall was 0.78, and F1-score was 0.77 (Zhou et al., 2020).

Furthermore, Vijayaraghavan and Guan (2022) utilized the data level approach to address the issue of class imbalance. The authors proposed using two different data augmentation methods to upsurge the size of the training dataset and improve classification results: random oversampling and GAN-based data augmentation. With random oversampling, they simply duplicate existing samples from the minority class to balance the dataset (Mqadi et al., 2021). With GAN-based data augmentation, they train a GAN to produce synthetic examples of the minority class that are more diverse and representative of real-world examples. By using these data level approaches, they are able to address class imbalance and increase the efficiency of learning models. The first method they use is random oversampling. This involves duplicating existing samples from the minority class to balance the dataset (Mqadi et al., 2021). For example, if there are only 100 positive samples and 900 negative samples, they would duplicate some of the positive samples to create a new dataset with 500 positive and 900 negative samples. While this method is simple and easy to implement, it can lead to overfitting because it does not add any new information about the minority class (Vijayaraghavan & Guan, 2022).

Fan et al., (2016) proposed a data level approach to address the issue of class imbalance in pattern recognition. The authors note that conventional data level methods are often used to preprocess datasets and balance the classes before training a classifier. However, these methods have limitations, such as the loss of information and increased computational complexity (Fan et al., 2016). To overcome these limitations, the authors propose a new approach called One-sided Dynamic undersampling No-Propagation Neural Networks (ODUNPNN). The approach involves dynamically undersampling the majority class during training, which reduces the computational complexity and preserves information. Additionally, the authors use a one-sided neural network architecture that only propagates information from hidden layers to output layers, further reducing computational complexity (Fan et al., 2016). In terms of references, the authors cite several studies that have addressed class imbalance in pattern recognition using data level approaches. For example, they cite Derrac et al., (2015), who developed the Keel software tool for data mining and experimental analysis. They also cite several studies that have used undersampling techniques to balance datasets, such as Batista et al., (2004). Overall, the article provides a detailed discussion of data level approaches for addressing class imbalance in pattern recognition and proposes a new approach that overcomes some of the limitations of conventional methods. The authors support their claims with experimental results and references to related studies in the field. The performance measures used in this article include G-Mean, AUC, and F1-score. The results of the experiments show that ODUNPNN outperforms in terms of AUC, G-Mean, and F1-score. Specifically, on binary-class imbalance datasets, ODUNPNN achieves an average AUC value of 0.962, which is higher than other compared approaches such as NPNN (0.947), RUS-NPNN (0.951), OSS-NPNN (0.952), LASVM-AL (0.954), EasyEnsemble (0.956), AdaBoost (0.957), and 1-NN (0.946) (Fan et al., 2016). Similarly, on multi-class imbalance datasets, ODUNPNN achieves an average AAUC value of 0.874, which is higher than other compared methods such as SMOTEBoost (0.862) and AdaBoost.M2 (0.864) (Fan et al., 2016).

Further, Han et al., (2005) proposed a data-level approach to address the problem of imbalanced data sets. The authors introduced two new minority oversampling methods which only oversample minority instances close to the borderline (Wan et al., 2017). The authors explain that traditional oversampling methods such as SMOTE produce fake examples by inserting among existing minority examples. The proposed borderline-SMOTE approaches report this issue by focusing on the minority instances that are near to the decision boundary between the minority and majority classes (Han et al., 2005). The performance measures used in this article include precision, recall, and F1-value (Wan et al., 2017). These measures are commonly employed to assess classification performance in imbalanced data sets (Han et al., 2005). Further, the results of the experiments showed that both borderline-SMOTE2 and borderline-SMOTE1 outperformed SMOTE and random over-sampling in terms of performance measures. For example, on the "ecoli" data set, borderline-SMOTE1 achieved a TP rate of 0.8 and an F1-value of 0.57, while SMOTE attained a TP rate of 0.6 and an F1-value of 0.43. In addition to TP rate and F1-value, the authors also reported other performance measures such as precision, recall, FP rate, and AUC. These measures provide additional insights into the classification performance of the proposed methods. For example, on the "yeast" data set, borderline-SMOTE1 achieved a precision of 0.5 and a recall of 0.67, while SMOTE attained a precision of 0.33 and a recall of 0.5.

Moreover, Singh et al., (2022) pointed that data augmentation involves generating synthetic examples by applying transformations to existing data points. Further, the authors argued that data augmentation can generate more synthetic samples for the minority class by applying transformations to existing data points which can help the classifier learn more features and patterns of the rare class. Furthermore, Langevin *et al.* (2022) argued that data augmentation can be used to generate more synthetic samples for the minority class. The proposed method is called majority-minority GAN transfer, which involves training GAN on the majority class and then using it to generate synthetic samples for the minority class (Zhou et al., 2020). Further, Langevin et al., (2022) argued that data augmentation can increase the size of the training dataset by generating new synthetic samples that are similar to the existing ones. This is particularly useful for imbalanced datasets where the minority class has very few samples compared to the majority class (Langevin et al., 2022). This can help balance out the dataset and prevent overfitting on the majority class (Langevin et al., 2022). However, Vijayaraghavan and Guan (2022) pointed out that the use of data augmentation can reduce the risk of overfitting in machine learning models by increasing the diversity and size of the training dataset. By doing so, they are able to increase the size of the training dataset and provide more examples for the machine learning algorithm to learn from (Sethia et al., 2018). This helps to reduce overfitting by preventing the model from memorising specific examples in the training set and instead learning more generalizable patterns that can be applied to new data. Additionally, by comparing GAN-based data augmentation with vanilla oversampling, they are able to evaluate which method is more effective at reducing overfitting and improving classification results (Awoyemi et al., 2017).

Further, the sampling methods are generally based on oversampling, undersampling, or the combination of both oversampling and undersampling techniques to deal with the imbalanced class challenge. The majority class represents valid transactions and the minority represents invalid or fraudulent transactions. In production environments, the majority of transactions made are legitimate, while a small fraction consists of invalid or fraudulent activity. Oversampling methods generate more balanced data by reproducing samples from minority groups. The data-level sampling techniques are used to adjust either by decreasing the samples of the majority class or by increasing the samples of the minority class. Generally, the outcomes of sampling techniques alter the distribution of datasets until it becomes balanced. The literature has shown that the balanced datasets can enhance the ability of the classifier. For instance, the SMOTE technique, or SMOTE, introduced by Chawla et al., (2002) is an intelligent data-level approach which adds artificial data points in the minority instances. In SMOTE, the minority class is oversampled by generating artificial examples instead of by using the replacement approach. In this approach, the minority class is oversampled by taking each minority sample as well as by introducing artificial examples besides line-segments joining the k minority class nearest neighbours, while the B-SMOTE (Chen et al., 2018) technique is a modified version of SMOTE (Ullah and Mahmoud, 2021). It pin-points the exact boundary between each class to improve the predictions. In addition, He et al., (2008) introduced an adaptive learning technique, ADASYN, for imbalanced data classification challenges. The technique has the ability to adaptively produce artificial data instances for the minority class

to lessen the bias due to imbalanced data distributions. Moreover, this algorithm shifts the classifier decision boundary to be more focused on those difficult to learn samples, thus enhancing the learning ability. In more recent years, many GAN-based techniques have emerged to deal with the imbalanced data, such as Vanilla GAN (Goodfellow et al., 2014), NS GAN (Shannon et al., 2020), CGAN (Mirza & Osindero, 2014), SDG GAN (Charitou et al., 2021), LS GAN (Mao et al., 2017), and WGAN (Arjovsky et al., 2017). Many way outs have been offered by these techniques at the data level and algorithmic level. At the data level, many GAN-based sampling frameworks are used to generate synthetic data to rebalance the dataset.

Furthermore, Lamba (2020) in their research work, introduced a Sparse Auto Encoder (SAE) and GAN-based model to differentiate fraudulent credit card transactions from non-fraudulent credit card transactions. This model is unique because it can be treated as a one class classification technique since it does not need mixed-type data sets comprised of negative and positive instances. The authors argue that cardholders have varying behavioural patterns while conducting monetary transactions via cards, so it may become hard to extract anti-fraud patterns. On the other hand, deep learning methods offer novel ways for detection. Therefore, in their empirical work, (Chen et al., 2018) attempted to apply a sparse autoencoder for separating fraudulent and non-fraudulent transactions. Ullah and Mahmoud (2021) presented optimal ways to identify financial frauds. The aim of their work is to discuss telecom fraud. The reasons behind this fraud are a lack of private information privacy, sloppy banking regulations, shortcomings in telecom supervision, the low rate of the detection of fraud cases, and identity theft. To address these challenges, authors have proposed a novel framework and named it an “Adversarial Deep De-Noising Auto-encoder” for detecting Telecom fraud at the receiving bank. It is noteworthy here that this novel approach is based on GAN. The proposed approach employs a deep denoising auto encoder to control noisy inputs and incorporates two high-end classifiers (for classification and discrimination) to boost learning efficiency. The findings of this empirical study reveal that this proposed model has significant rewards in terms of the misclassification rate and the sound classification accuracy than other state-of-the-art approaches. In addition, this anticipated framework was applied to two conventional banks and effectively detected 321 fraud cases.

To conclude, this approach successfully lessened customer losses and enhanced the reputation of commercial banks. Recently, Li et al., (2021) conducted a study on cyber-attacks. The authors argue that fraudsters are using unique and novel methods to conduct cyber-attacks. They emphasize that deep machine learning techniques have convinced researchers by detecting anomalies effectively. They argue that neural networks are excellent substitutes for the detection of anomalies. In their study, the authors introduced an anomaly-based intrusion method for IoT networks. They implemented their model with the help of neural networks (NN) in 1D, 2D, and 3D. In their study on data-level algorithms, Grandini et al., (2020) argued that credit card-based fraud has become the biggest cyber-based fraud faced by cardholders. To mitigate such fraudulent activities, deep machine learning-related detection systems are a better option.

Nevertheless, designing machine learning methods is challenging due to problems associated with class imbalance in datasets. Data level augmentation techniques are effective in addressing the challenges arising due to imbalanced data. They can reduce bias by generating new samples of minority classes which can be used while training a classifier. However, they require careful tuning and selection of parameters to ensure that the generated samples are appropriate for training and to enable better performance on unseen data. Therefore, further research is needed to explore the effectiveness of these techniques for different datasets and applications. In more recent times, GANs have gained immense success in various domains. Many researchers have contrasted GANs for imbalanced data scenarios against other well-known methods. It is imperative to mention that the detection of fraudulent transactions is an expensive and time-consuming task. In the past, unsupervised methods have been proposed to deal with this challenge. Data level augmentation approaches methods such as GANs have the ability to simulate high dimensional and complex data distributions can be employed to learn the behavioural instances of normal data to detect anomalies. These developments in GAN are making it the most effective method. However, more research work is needed in order to improve the predictability, efficacy, accuracy, and applicability of GAN variants. In this study, we concentrate on data augmentation approaches.

2.3.1. Sampling Based Techniques

In this study, we consider existing data augmentation techniques, i.e., Vanilla GAN (Goodfellow et al., 2014), NS GAN (Shannon et al., 2020), CGAN (Mirza & Osindero, 2014), SDG GAN (Charitou et al., 2021), LS GAN (Mao et al., 2017) and WGAN (Arjovsky et al., 2017) as well as sampling based techniques have been ADASYN (He et al., 2008), SMOTE (Chawla et al., 2002), B-SMOTE (Han et al., 2005) and our proposed K-CGAN. To conduct our experiments we utilised Python, Jupiter notebook, and Tensorboard. We have utilised the following libraries, such as Tensorflow, as our Machine Learning Framework, and required layers to define a neural network such as Input, Embedding, Dense, Dropout, Flatten, Activation, Reshape, Concatenate from tensorflow.keras.layers library, and further libraries such as numpy, pandas, sklearn.preprocessing, min.maxscaler, seaborn, sys, time, SMOTE, ADASYN, BorderlineSMOTE from imblearn.over_sampling, roc_curve from sklearn.metrics, stats from scipy, LogisticRegression from sklearn.linear_model, GaussianNB from sklearn.naive_bayes, KNeighborsClassifier from sklearn.neighbors, RandomForestClassifier from sklearn.ensemble, xgb from xgboost, os, norm from numpy.linalg, plt from matplotlib.pyplot, PCA from sklearn.decomposition, Axes3D from mpl_toolkits.mplot3d, ArgumentParser from argparse, and train_test_split from sklearn.model_processing.

2.3.1.1. SMOTE

Since its introduction in 2002, SMOTE (Chawla et al., 2002) has been successfully used in a wide range of contexts and fields. The development of several distinctive supervised training paradigms, such as incremental learning, multi-label classification, multi-instance learning, and semi-supervised learning, has been influenced by SMOTE, which seeks to overcome the issue of class imbalance. The method is unequalled for learning from different data sources. SMOTE performs better when the dataset size is small. However, if the size of the dataset is large, SMOTE takes time to create artificial data points, and SMOTE's efficiency drops significantly. Furthermore, while creating artificial data points, the chance of overlapping data points for the minority class is high in SMOTE. In the first step of basic principle of SMOTE, each sample x_i is selected in turn from the minority samples as the root sample for the synthesis of the new sample. After that, the process randomly selects a sample from the K neighbour samples of the same category of x_i as the auxiliary sample for the synthesis of new sample, the process is repeated n times. After that, linear interpolation between the sample x_i and each auxiliary sample through Eq. (1), and at last n synthesized samples are generated.

$$x_{new,attr} = x_{i,attr} + rand(0,1) \times (x_{ij,attr} - x_{i,attr}) \quad (1)$$

Among them, $x_{i,attr}$ is the *attr* attribute value of the i sample in the minority class. Moreover, $rand(0,1)$ is a random integer between [0,1]. $x_{ij,attr}$ is the j nearest neighbor sample of sample x_i . Whereas, x_{new} is a new sample synthesized between x_{ij} and x_i . The Eq. (1) demonstrates that the new sample x_{new} is a sample derived from the interpolation between x_i and x_{ij} .

2.3.1.2 ADASYN

ADASYN (He et al., 2008) is used to create minority data samples with distributions that reflect those of the underrepresented groups with the goal of generating more data to address the data imbalance. ADASYN has the ability to generate data samples for minority class samples which are hard to learn. Furthermore, the generated data points using ADASYN (Chen et al., 2018) not only balance the dataset well but also reduce the learning bias of the actual dataset. Additionally, this method is also applicable for the multiple-class imbalanced learning challenge. On the other hand, the major drawback of this algorithm is that ADASYN's precision may suffer due to the nature of adaptability. In addition, each of the neighbourhoods only contain

one minority example for minority samples which are sparsely distributed. Further, both SMOTE and ADASYN have a common ancestor. However, ADASYN introduces a tiny random bias to the points after the samples are formed, making them less closely related to their parents. The variance of the synthetic data is increased although this is a small adjustment. To provide fake information, a synthetic adaptive algorithm is used to create minority data samples that have distributions that are typical of the underrepresented groups in order to address the data imbalance.

$$s_i = x_i + (x_{z_i} - x_i)\lambda \quad (2)$$

Whereas, minority cases x_i and x_{z_i} in the same neighbourhood as the innovative synthetic example are generated using a random integer between 0 and 1. Moreover, x_{z_i} is minority sample from the K nearest neighbors of x_i . In addition, s_i is synthetic data example that is generated according to Eq. (2), among them λ is a random number $\lambda \in [0,1]$.

2.3.1.3 B-SMOTE

A Borderline-SMOTE (Han et al., 2005) only generates synthetic instances for the minority occurrences that are close to the boundary of two categories. In the majority of classification systems, researchers used B-SMOTE (Chen et al., 2018; Han et al., 2005) during training to pin-point the exact boundary between each class to improve the predictions.

$$P = \{P_1, P_2, \dots, P_{pnum}\}, N = \{n_1, n_2, \dots, n_{nnum}\} \quad (3)$$

In the Eq. (3), the entire training set is assumed as T i.e., both P and N . The entire minority class is supposed as P and the majority class is assumed as N . Moreover, $pnum$ represents the total number of minority instances, and $nnum$ represents the total number of majority cases. Moreover, for each $P_i \{i = 1, 2, 3, \dots, P_{pnum}\}$ in the minority class P , its m nearest neighbors are calculated from the entire training set T .

2.3.2. GAN-Based Techniques

2.3.2.1 Vanilla GAN

Goodfellow et al., (2014) introduced Vanilla GAN as a method to generate data from a given set of observations. In order to produce new data, the generator G first seeks to identify the distribution within the training data. The discriminator has been trained to output the likelihood that the input data is derived from noise from the generator or the training set. In order to trick the discriminator into classifying the data it creates as the training set data, the generator seeks to provide data that is slightly closer to the training dataset (Langevin et al., 2022).



Figure 1: GANs Architecture

The architecture of a GAN, commonly used, is depicted in the graphical schematic representation shown in Figure 1.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data(x)}} [\log D(x)] + E_{z \sim P_Z(z)} [\log (1 - D(G)(z))] \quad (4)$$

To learn the distribution P_g over data x , Generator G builds a mapping function from noise distribution $P_Z(z)$ to data space. Moreover, the Discriminator D outputs a single scalar signifying the probability that x came from training data instead from P_g .

Where, $E_{z \sim P_Z(z)}$ is a random noise (usually standard Gaussian) and $E_{x \sim p_{data(x)}}$ is the true data distribution. In addition, parameters are adjusted for D to $[\log D(x)]$ and parameters are adjusted for G to minimize $[\log (1 - D(G)(z))]$ as both D and G follow min-max game with value function $V(D, G)$.

2.3.2.2 CGAN

GANs can be extended to conditional frameworks by conditioning both the G and D on additional information. A type of GAN augmentation known as CGAN (Mirza and Osindero, 2014), takes into account extra limitations. To satisfy this requirement, the discriminator and the generator must both consider a third piece of information, denoted y . This third piece of information might be anything from data from a different domain to a classifier.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data(x)}} \left[\log D\left(\frac{x}{y}\right) \right] + E_{z \sim P_Z(z)} \left[\log (1 - D(G\left(\frac{z}{y}\right))) \right] \quad (5)$$

Here, D represents the discriminator, G represents the generator, y and $P_Z(z)$ as input noise are combined in the G in joint hidden representation. Eq. (5) presents mathematical representation of conditional GAN which consists of two adversarial models: a generative model G that captures data distribution and a discriminative model D that calculates the probability that a sample is from the training data rather than G . In order to learn a generator distribution P_g over x , the G build a mapping function from a noise distribution $P_Z(z)$. Moreover, the D outputs a single scalar signifying the probability that x came from training data rather than P_g . In addition, both G and D are trained simultaneously. Moreover, GANs function is extended in Eq. (5) to a conditional model where both G and D are conditioned on some extra information y . Whereas, y is auxiliary information, for instance class labels. In cGANs, conditioning is performed by feeding y to G and D as additional input layer. Moreover, parameters are adjusted for G to minimize $\left[\log (1 - D(G\left(\frac{z}{y}\right))) \right]$ and parameters are adjusted for D to minimize $\left[\log D\left(\frac{x}{y}\right) \right]$ as both these neural networks follow min-max game with value function $V(D, G)$.

2.3.2.3 WGAN

To encourage effective training, this architecture alters the loss role of the default application and uses a weight clip (Arjovsky et al., 2017). They propose to compute the loss function using the earth mover distance rather than the Jensen–Shannon divergence. This reserve metric, which assesses the similarity of the data distributions from the exercise dataset and the created dataset, is constant and observable throughout.

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y) \sim \gamma} [\|x - y\|] \quad (6)$$

Eq. (6) presents the earth mover distance also known as Wasserstein-1. Among them, $\Pi(P_r, P_g)$ is the combination of all distributions $\gamma(x, y)$ whose marginal are P_r and P_g , respectively. Moreover, $\gamma(x, y)$ demonstrates the quantity of mass must be shifted to y from x to transform P_r distributions into P_g distributions. By doing so, the earth mover distance will be the cost of most desirable transport plan.

2.3.2.4 SDG GAN

The generator and discriminator of the SDG GAN (Charitou et al., 2021) are both convolutional channels with an MLP design in the SDG GAN framework. A standard GAN's generator seeks to produce fake data that closely resemble the true distribution (Taha and Malebary, 2020). Synthetic Data Generation GAN has the capacity to outperform density-based oversampling methods and enhances the classification ability of benchmark datasets and real fraud datasets.

In addition, original GAN uses regular loss, however, instead of using regular loss, the SDG GAN utilizes feature matching loss to improve GAN training. The feature matching objective function is shown in Eq. (7).

Moreover, SDG GAN used conditional GAN method to calculate the $P_{x/y}^x$, which denotes conditional distribution. The conditional distributions allow us to sample the minority class by conditioning the G on the minority class label, which is presented by $X_{new} = G(z, y = y^{minority})$. In order to reduce the statistical disparities between the characteristics of the actual data and the produced data, feature matching adjusts the objective functions for the generator. As a result, the generative network's focus shifts from deceiving the adversary to matching features in the actual data. On the other hand, D is trained same as to a regular GAN discriminator. The empirical findings in original SDG GAN paper indicate that feature matching is an effective approach where a regular GAN suffers instability. The original SDG GAN achieved the following objective function:

$$\min_G \max_D \|E_{z \sim p_{data}} f(x/y) - E_{z \sim p_z(z/y)} f(G(z))\|_2^2 + E_{x \sim p_{data}} [\log(D(x/y))] \quad (7)$$

In Eq. (7), the term $\|E_{z \sim p_{data}} f(x/y) - E_{z \sim p_z(z/y)} f(G(z))\|_2^2$ is the feature matching loss. Moreover, the rest of objective function is binary cross entropy, or simply BCE between true class labels $y \in (0, 1)$ and the predicted class probability.

2.3.2.5 NS GAN

Non-saturating GANs (NS GAN) (Shannon et al., 2020) is an improved version of GANs (Goodfellow et al., in 2014). The form of GAN that is most widely used as a benchmark in research and practical applications is non-saturating GAN (NS GAN). However, the NS GAN algorithm lacked theoretical justifications (Taha and Malebary, 2020), like other GANs such as WGAN. The loss function performs poorly in practice; despite

being outstanding for theoretical results. The GAN has difficulty converging, stabilising its training, and offering a range of samples. The aforementioned loss function for G should not be trained, but rather improved gradients from prior training should be utilized (Taha and Malebary, 2020).

In the adversarial game, the D tries to classify which data is fake or real. On the other hand, the objective of G is to fool D into thinking that the fake data it produces is real data. According to the original paper, in which GAN was introduced, maximizing the objective of D after swapping data (NS GAN) gives better results than minimizing the objective of D (Saturating GAN) directly. The mathematical representation of NS GAN can be shown as the following two steps:

$$\max_{D: X \rightarrow [0,1]} E_{x \sim P} [\log(D(x))] + E_{z \sim P_Z} [\log(1 - D(G(z)))] \quad (8)$$

$$\max_{G: X \rightarrow [0,1]} E_{z \sim P_Z} [\log(D(G(x)))] + E_{x \sim P} [\log(1 - D(x))] \quad (9)$$

In Eq. (8) and (9), P is the real data distribution on domain X and Z is the domain of P_Z . It is noteworthy to mention that the fake data distribution is formed by $G(z)$ as Q .

Moreover, Eq. (10) and Eq. (11) are generalized in the following matters:

$$\max_{D: X \rightarrow \text{dom} f} E_{x \sim P} [f(D(x))] + E_{z \sim P_Z} [g(D(G(z)))] \quad (10)$$

$$\max_{G: Z \rightarrow X} E_{z \sim P_Z} [f(D(G(x)))] + E_{x \sim P} [g(D(x))] \quad (11)$$

Whereas, both f and g are scalar-to-scalar functions selected in order to predict the chances of the data being real by the D . Generally, f is a monotone increasing and on the other hand, g is monotone decreasing.

2.3.2.6 LS GAN

The foremost benefit of LS GANs (Mao et al., 2017) is that unlike conventional GANs, where there is nearly no loss for samples that lie on the correct side of the decision boundary, LS GANs can penalize samples although they are rightly classified. The other benefit is that the decision boundary can produce more and more gradients when updating the G , this then lessens the issue of the vanishing gradient (Sohony et al., 2018). This design shows that the model trains more steadily and is better able to handle the gradient vanishing problem than the vanilla method by using the least square error as the loss. The aim of G is to learn the distribution P_g over data x . The G begins from sampling input variables z from a Gaussian distribution $P_Z(Z)$ then maps the input variables to data space $G(z)$ via a differentiable network. In addition, D is a classifier $D(x)$ that tries to recognize the instance from training data or from G .

$$\min_D V_{LSGAN}(D) = \frac{1}{2} E_{x \sim \text{pdata}(x)} [(D(x) - b)^2] + \frac{1}{2} E_{z \sim P_Z(Z)} [(D(G(z)) - a)^2] \quad (12)$$

$$\min_G V_{LSGAN}(G) = \frac{1}{2} E_{z \sim P_Z(Z)} [(D(G(z)) - c)^2] \quad (13)$$

Where b is the label for real data, a for fake data, and c is used as a label for testing the discriminator D with generator G samples(x) i.e. $b = 1$, $a = 0$, and $c = 1$.

2.4 Classifiers

Five popular classification techniques XGBoost (Chen and Guestrin, 2016), Random forest (Breiman, 2001), K-Nearest Neighbor (Cover and Hart, 1967), Multilayer Perceptron (Rosenblatt, F., 1957), and Logistic regression (DeMaris, A., 1995) are implemented to evaluate the performance of the data augmentation methods in this study.

2.4.1 XGBoost

XGBoost (eXtreme Gradient Boosting) is a widely used machine learning technique, developed by (Chen and Guestrin, 2016). This technique improves the initial gradient-boosting technique. By using ensemble techniques, it improves functionality in general. To solve the issue of a non-uniform majority class, researchers modify traditional classification algorithms utilizing ensemble approaches. To complete a categorization exercise as an “ensemble,” a group of students is gathered. The performance of a classifier is increased by combining numerous weak learners into a small number of robust ones (Hajek et al., 2022). The loss functions in which 't' iteration needs to be minimized is given below:

$$L^{(t)} \sum_{i=1}^n l(y_i, \hat{y}^{t-1} + f_t(x_i)) + \Omega(f_t) \quad (14)$$

The XGBoost model is trained in an ‘addictive’ way. Where, l is a differentiable convex function that computes the difference between \hat{y}^{t-1} and y_i . Moreover, Ω penalizes the complexness of the regression tree functions. It is noteworthy to mention that y_i is the real label known as the training dataset, whereas l is a function of CART learners which is sum of the previous and current addictive trees. In addition, \hat{y}^t is the prediction of i -th instance at t -th iteration. The introduction of f_t is to minimize the objective function.

2.4.2 Random Forest

Random Forest (Breiman, 2001) is a supervised machine learning technique that can be used to solve regression and classification issues (Sadgali et al., 2019). It builds several Decision Trees during training and employs a majority vote to decide the outcome in order to improve accuracy and produce more reliable forecasts. To increase precision, Bootstrap aggregation and entropy criteria are applied.

The most common element in all the processes is that the k -th tree, a random vector θ_k is produced, which is independent of the previous random vectors $\theta_1, \dots, \theta_{k-1}$, however, having the same distribution. Moreover, tree is grown using the training set and θ_k resulting in a classifier $h(x, \theta_k)$ in which x is an input vector. For example, in bagging the random vector θ is produced as the counts in N boxes as a result from N darts thrown at box randomly, in which N is number of instances in the training set. In random split selection θ comprises of a number of independent integers from 1 to K . The nature of θ depends on its utilization in tree construction. In addition, when multiple trees are created, they pick the most well-known class, which are known as Random Forests (Breiman, 2001).

Moreover, a Random Forest is a classifier having a collection of tree like classifiers $\{h(x, \theta_k, k = 1, \dots)\}$ where the $\{\theta_k\}$ are independent identically distributed random vectors and each of the tree case a unit vote for the well-known class at input x .

2.4.3 K-Nearest Neighbor

The K-Nearest Neighbor introduced by (Cover and Hart, 1967) is a supervised method widely adopted within machine learning. KNN methods choose an integer k that separates the data from the closest neighbours again (Makki et al., 2019). Its principal usage is the classifying process. The similarity of a new data point to previously classified data affects its classification. Integer k is selected by KNN algorithms to once more split the data from its nearest neighbours.

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (15)$$

The distance between points is calculated using a specific norm. The class with most of the K closest points is given to the new observation. An observation is defined as R_n , and the norm is typically employed to calculate the distance among 2 observations, q and p .

2.4.4 MLP

The Multi-layer Perceptron (MLP) was first introduced by (Rosenblatt, F., 1957), several decades later MLP was used to develop ANNs. A multi-layer perceptron is a synthetic system with at least three layers of nodes (hidden, input and output). Each node makes use of an encoder. The activated function adds bias after computing the weighted sum of its inputs. This allows researchers to select which transistors have to be removed and ignored while making outside networks (Lamba, 2020).

$$x_f^2 = \frac{12N}{K(K+1)} \left[\sum R_j^2 - \frac{K(K+1)^2}{4} \right] \quad (16)$$

Where K stands for the total set of algorithms, N stands for the number of data sets, and R_j^2 represents the algorithm j 's average rank.

2.4.5 Logistic Regression

Logistic regression, as its name suggests, is a kind of regression model that makes use of a categorical dependent variable (DeMaris, A., 1995). Using logistic regression, one or more independent variables can be used to estimate the probability of a binary response. Forecasts are transformed into probabilities using the sigmoid function (Wang and Zhao, 2022). Linear regression's loss function is squared loss. Similarly, the logistic regression's loss function is log loss, which can be defined as follows:

$$\log \text{loss} = \sum_{(x,y) \in D} -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \quad (17)$$

Among them $(x, y) \in D$ is the dataset which contains multiple labelled samples which are (x, y) pairs. In addition, y is the label in labelled example. In logistic regression, every value of y must be either 0 or 1. Moreover, \hat{y} is predicted value which must be anywhere between 0 and 1, given the set of features in x .

2.5 Overview of State-of-the-art Methods

The diagram in Figure 2 illustrates various methods for handling imbalanced classification issues.

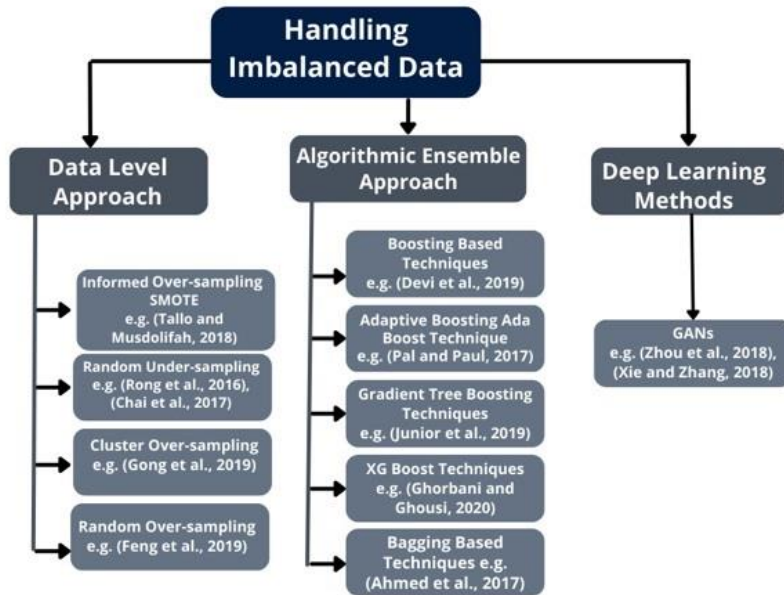


Figure 2: Techniques to handle imbalanced classification issues

To provide justification for using GAN over other existing approaches, we conducted an analysis of various techniques. Below are the pros and cons of these techniques, based on which we chose GAN for comparison.

Names of the techniques previously used:

- 1) Resampling, which includes oversampling and undersampling.
- 2) SMOTE.
- 3) Ensembling methods.
- 4) GAN based methods.

Ensemble techniques are techniques that improve the accuracy of results in models by combining many models rather than relying on just one. The precision of the results is significantly increased by the integrated models. Therefore, ensemble methods for machine learning have become more popular (Sohony, Pratap and Nambiar, 2018).

2.5.1 Resampling (Oversampling and Undersampling)

According to (Amin et al., 2016), resampling has key features as follows:

- 1) It relies on the assumption of a known or reasonably approximated population size. For example, if researchers aim to study the size of rodents in a specific region, an estimation of the rodent population is necessary to determine a starting point or sample size effectively.
- 2) Resampling requires a certain level of randomness in the population. When the population exhibits a consistent pattern, there is a higher risk of inadvertently selecting cases that are excessively regular.

To address data imbalance, common techniques are oversampling and undersampling. Downsampling reduces data for majority classes, while oversampling increases data for minority classes. Downsampling, however, may result in the loss of important information about the majority classes, while oversampling without careful consideration can lead to overfitting.

2.5.2 SMOTE

The position of the minority class has an impact on SMOTE techniques. When classifications overlap or there is subtraction in the data, SMOTE reveals issues. If classes overlap inside a bigger cluster, this is known as a disjunction (Prati et al., 2004). This implies that the method may generate more data in a region that is difficult to separate and needs more sophisticated classifiers. This is taken into consideration by cluster-based SMOTE expansions, such as ADASYN, but it is limited by presumptions. According to Cordón et al., (2018), it's possible that these presumptions do not apply to producing these complicated distributions. Further, SMOTE technique is unparalleled for learning from many data sources. At smaller dataset sizes, SMOTE works remarkably well. SMOTE's efficiency decreases noticeably when the dataset is huge since it takes a long time to generate fake data points. Also, the likelihood of overlying data points for the minority class in SMOTE is significant when constructing false data points. It is observed that the sensitivity of SMOTE is lower as compared to other GAN based models (Fiore et al., 2019). However, SMOTE generates examples that are similar to existing ones and related to a learning algorithm, the newly created instances are not exact duplicates. Resultantly, the decision boundary is softer as an outcome.

Nevertheless, when constructing the synthetic examples, SMOTE ignores the placements of the nearby instances from other classes (Fiore et al., 2019). Moreover, SMOTE tries to manipulate the instances mindlessly without taking the data distribution into account, leading to the creation of incorrect instances that end up in the majority of regions when there is class dissemination or class noise (Cao et al., 2017). Furthermore, SMOTE misclassifies several majority models as a minority, lowering SVM classification efficiency, particularly for complicated classification problems resulting from complicated distributions (Cao et al., 2017). For instance, class overlap, whereas the data distribution becomes more distinguishable in the kernel space by locating with a suitable kernel matrix, helping to ensure that the instances created seem to be more precise than oversampling in the original feature space. This illustrates that the premise that features space is a better place for oversampling than input space is correct. Through transferring with a suitable kernel matrix, the distribution of the data reduces the possibility of class overlapping in feature space, guaranteeing that the instances created are more precise than the frame interpolation in the original space (Cao et al., 2017). Furthermore, there is a notable rise in the concentration of artificial minority samples near prominent examples of the challenging minority class, compared with earlier examples of the same (Sun et al., 2020). As a countermeasure, SMOTE will probably have fewer synthetic minority samples surrounding the easy minority class samples that were properly categorised by the previous base classifier. Moreover, there is a smaller drop-off in the frequency of synthetic minority samples surrounding more recent easy minority class samples compared to those surrounding earlier easy minority class samples (Sun et al., 2020).

Further, He and Garcia (2009) pointed out that one potential drawback of SMOTE seems to be that noisy samples might be created due to the difficulty in distinguishing between minority and majority class clusters. Furthermore, it is observed that SMOTE generates inappropriate accuracy for class imbalance data (Douzas and Bacao, 2018). In place of sample replication, cutting-edge oversampling techniques such as SMOTE (Chawla et al., 2002), produces synthetic training instances from the minority class using interpolation. Such approaches have some important downsides. For instance, the possibility of oversampling and undersampling leading to overfitting (Chawla et al., 2002). Further, it is examined that when SMOTE is employed, the precision measure significantly decreases (Charitou et al., 2021). Though the ability to produce synthetic data makes oversampling approaches, such as the SMOTE, a popular choice among academics. Newly generated minor samples might overlap with existing major samples, which is one of SMOTE's major drawbacks. This raises the risk that machine learning techniques will exhibit bias when it comes to the way they perform for major class increases (Manjurul Ahsan et al., 2022). Nevertheless, conventional SMOTE generates additional noise and cannot deal with high-dimensional data (Maldonado et al., 2022; Wang et al., 2021). Further, the SMOTE technique preprocessing is helpful in identifying the decision boundary because of the imbalance distribution of user features, but still, it takes more time to build the appropriate model on a substantial number of virtual datasets.

In addition, the placement of the minority class has an impact on SMOTE algorithms. When there are disparities in the data or overlap between classes, SMOTE encounters challenges. These sections of a larger cluster, where classes intersect, are known as disjoints (Prati et al., 2004). Consequently, the algorithm might generate additional data in areas that are hard to differentiate and necessitate more advanced classifiers. Due to its unpredictability, the random oversampling-based approach, which repeats certain samples, raises the total number of minority classes but is unable to effectively retrieve the samples close to the border between the minority and majority classes (Zhang et al., 2019). Focusing on a few criteria, SMOTE-based oversampling algorithms frequently extend the data to several class samples. These techniques circumvent the problem of sampled data but are still unable to lower the rate of false sample overlaps (Ni et al., 2023).

Nevertheless, the issue of overgeneralization in oversampling approaches, and particularly for the SMOTE algorithm, is primarily linked to the method used to construct synthetic samples. SMOTE properly creates the same number of the synthesised dataset for each actual minority instance without taking into account nearby instances, which enhances the chance of class overlap (Wang and Japkowicz, 2004). Further, Chawla et al., (2002) argued that one of the most popular and productive techniques is that SMOTE uses k Nearest Neighbors for every minority sample to produce synthetic data based on the shared characteristics between the minority class samples. The primary drawback of this method is that the samples of simulated data may intersect with the data of the majority class (Chawla et al., 2002). Moreover, the cornerstone of the SMOTE method is resampling all along sample points that intersect each sample from the negative class and its designated nearest neighbours, and thus can successfully prevent the unintended consequence of the random oversampling approach (Xue and Zhang, 2016). Nevertheless, SMOTE overlooks the distribution of the positive class samples close to the negative samples in the phase of generating negative class data because it assumes that the neighbours still pertain to the same class. There is a certain opacity in the closest neighbour selection technique since it does not take the training set's features into account. Furthermore, the SMOTE is the primary oversampling method that aims to normalize the dataset by adding a few fictitious minority samples (Chawla et al., 2002). Minority samples are created using SMOTE and its variations (Das et al., 2014; Han et al., 2005; Lim et al., 2016; Zhang and Li, 2014) by spontaneously interpolating between a minority sample and its closest minority neighbours. The primary flaw in these approaches is that they fail to take into account the dataset's initial distribution, which might lead to an inaccurate expansion of the minority location. Moreover, the SMOTE algorithm includes flaws such as variance and generalisation (Wang and Japkowicz, 2004).

Advantages and Disadvantages of SMOTE

As discussed in previous section, SMOTE has several advantages and disadvantages (Chou et al., 2020). Below we summarise the list:

Advantages:

- 1) It is relatively easy to develop and implement specific examples, making it suitable for studies or surveys with limited budgets.
- 2) It provides a sense of control and interaction for specialists and analysts, especially when dealing with precise limits or narrowly framed speculation.
- 3) It eliminates group selection issues.
- 4) It is a generally a safe factor, with a low probability of data being impaired.

Disadvantages:

- 1) SMOTE's method of indiscriminately summing up the minority class can lead to overgeneralization, especially in cases of highly skewed class distributions.
- 2) The number of synthetic samples generated by SMOTE is fixed in advance, lacking flexibility in the re-balancing rate.
- 3) SMOTE is not efficient for high-dimensional data.

2.5.3 ADASYN

The multiple-class imbalanced learning problem may also be solved using this strategy. On the other hand, the main disadvantage of this algorithm is that because of flexibility, ADASYN's (He et al., 2008) precision may deteriorate. Also, for the poorly dispersed minority samples, each of the neighbourhoods only has one minority case. Further, it is observed that ADASYN misclassifies several majority samples as a minority, reducing the SVM efficiency of the classifier, particularly for complicated classification problems stimulated by complex distribution (Cao et al., 2017). For instance, class overlap, whereas mapping with a suitable kernel matrix makes the data distribution more decomposes in the kernel space, making sure that the instances produced seem to be more precise than the oversampling in the original feature space. This proves that feature space is better for oversampling than input space. However, the distribution of the data reduces class overlapping in the training dataset by mapping with a suitable kernel matrix, making the instances created more precise than the oversampling in the original feature space (Cao et al., 2017). Furthermore, it is observed that ADASYN generates inappropriate accuracy for class imbalance data (Douzas and Bacao, 2018). Moreover, it is pointed out that ADASYN has a lower precision value and F-1 score as compared to CGAN (Mirza & Osindero, 2014), SDG GAN (Charitou et al., 2021), B-SMOTE (Han et al., 2005) and SMOTE (Chawla et al., 2002). However, Singh et al., (2021) pointed out that ADASYN generates lower recall and accuracy values as compared to other GAN based models.

2.5.4 Ensembling methods

Advantages:

Ensembling techniques have become effective tools with a number of advantages over conventional approaches.

- 1) According to Sharma et al., 2017, ensemble approaches outperform individual models like Support Vector Machines (SVM) and Artificial Neural Networks (ANN) in terms of classification effectiveness.
- 2) In comparison to previous AI approaches, ensemble learning models have been shown to exhibit improved prediction accuracy. As a result, they are appropriate for real-time applications such as system failure detection (Wang et al., 2018).
- 3) Machine learning models may efficiently minimize bias and variance by using ensemble neural networks, which results in overall error reduction. The ensemble's capacity to generalize is improved by this property, which also strengthens and improves the accuracy of the ensemble (Mehmood *et al.*, 2021).
- 4) According to (Ahmad and Brown, 2013) several ensemble approaches, such the new Random Projection Random Discretization Ensembles, have shown to be resilient to class noise.

Disadvantages:

- 1) Ensembling typically requires training and combining multiple models, leading to increased computational resources and time consumption. As the number of models in the ensemble grows, so does the complexity, making it less practical (Breiman, 1996).
- 2) Ensembling introduces additional hyperparameters that need to be tuned, such as the choice of base models, their weights, and the combination method (e.g., averaging, voting, stacking). Selecting the right combination can be challenging (Caruana et al., 2004).
- 3) Each individual model in the ensemble requires memory for storage, and when combining several models, the overall memory footprint increases. This could be a problem for resource-constrained environments (Fernández-Delgado et al., 2014)
- 4) They often create more complex models, making it difficult to interpret the combined model's predictions and understand the underlying decision-making process. The lack of interpretability can

be a concern in certain applications, especially those requiring transparency and accountability (Wolpert, 1992).

- 5) Although ensembling is often used to reduce overfitting, there is still a risk of overfitting if the individual models in the ensemble are themselves overfitting the training data. If the base models are too complex or trained on insufficient data, the ensemble may not generalize well to new and unseen data (Dietterich, 2000)
- 6) For ensembling to be effective, the individual models in the ensemble should have diverse strengths and weaknesses (Caruana et al., 2004).
- 7) In some cases, if the individual models in the ensemble are already strong performers, the marginal gain achieved by ensembling may not justify the additional computational cost and complexity. (Fernández-Delgado et al., 2014).

2.5.4 GAN based methods

Recently, GANs initially proposed by (Goodfellow et al., 2014), have attracted researchers to data generation due to their performance and elegant theoretical base (Xie et al., 2018). The purpose of generating synthetic data is to create data that can perform just like the real-world dataset for analysis tasks, for instance, classification. However, GANs have a few classification problems where it needs to consider class labels during synthetic data generation. To deal with the class label problem, researchers introduced GAN extensions. For instance, (Mirza and Osindero 2014) introduced Conditional GANs that consider class labels while generating new data. In their study, (Vega-Márquez et al., 2020) used Conditional GAN instead of conventional GAN to evaluate the practicality of generated data. The authors argue that Conditional GAN performs well in datasets with target class as they consider these details to train the network. In addition, a Conditional GAN-based framework generates synthetic data from the training data, which can be utilised for the same tasks without revealing the real data. Generally, researchers argue that training GAN framework is a difficult task (Mescheder et al., 2018). One of the main reasons is that the Generator has limited modelling capacity which prevents it from reproducing all nuances of the data. Also, the stability of GAN is a crucial concern while training GAN as it is challenging to balance GAN's component networks. If the Discriminator outperforms the Generator, the entire GAN training would not be efficient. Similarly, if the Discriminator performs poorly than its counterpart, the overall training results will be useless. Thus, in severe imbalancing state, the entire GAN framework needs to catch up to one component's failure against the other (Fiore et al., 2019).

Moreover, in their study, (Arjovsky and Bottou 2017) also points out that GANs are extremely hard to train, and researchers need more understanding of the instability of GANs while training. Many GAN extensions have also been proposed to counter this issue, but they rely on heuristics highly sensitive to modification. Thus, their limited applicability makes it challenging to test with new GAN extensions or employ them in new areas. The study by (Fiore et al., 2019) presented a GAN framework to resolve class imbalance in data. The study trained a GAN to generate only minority class instances and merged them with training data into an augmented training set to improve the classifier's effectiveness. Their empirical study observed that injecting artificial samples in the training set increases FPs. Thus, ineffective in setting where the FPs is costlier. Their framework depended on the availability of labelled instances of fraudulent credit card transactions. So, their proposed framework is observed to be ineffective in an unsupervised setting. Moreover, their GAN-based framework could not detect fraud patterns that are completely new in nature and have no details for generalisation. However, the authors emphasise that these shortcomings of their GAN framework are because the training was done on a small set of training data. There is a need for more research in this domain to check the effectiveness of their tuned GAN framework.

Recently, researchers used GANs to generate accurate synthetic data and tackle the issue of class imbalance (Efimov et al., 2020). GANs were able to generate accurate data samples. At the same time, GANs are hard to train. As GAN trains two networks concurrently, it means that when the parameters of a model are updated,

the problem of optimization changes. This framework setting constructs a dynamic system that is not easy to control. Non-convergence is a common problem while training GAN. Usually, deep learning methods are trained using an optimization algorithm that achieves the lowest point of a loss function. On the contrary, the game theory-based two-player situation may not reach equilibrium. Thus, the gradients may not converge and cannot attain the optimum minima. In other words, when the Generator gets too good, it may trick the Discriminator and can halt receiving critical feedback. As a result, the Generator will only receive bad feedback, leading towards output quality collapse (Eckerli and Osterrieder, 2021). In addition, mode collapse is the most common issue in the GAN framework while generating synthetic data and avoiding the problem of class imbalance. Mode collapse is a failure of GANs that occur because of training deficiencies. Mode collapse can happen when the Generator does not consider a region of the target data distribution. In other words, the Discriminator will learn to discriminate against the Generator's fake samples when the Generator upholds itself in a local minimum limited sample generation. Ultimately, this will end the learning phase and leads to an undiversified output (Eckerli and Osterrieder, 2021). This is a considerable issue as in generative modelling. The purpose is not only to generate realistic samples but also to be capable of producing a wide variety of samples (Takahashi et al., 2019).

Recent studies of GAN have demonstrated that when the Discriminator dominates the training process, the Generator can fall short. This failure is because an optimal Discriminator needs to give adequate feedback for the Generator to learn properly. This problem is known as the vanishing gradient problem, where the gradient gets too minimal that in back-propagation it does not change the weight values of the initial layers of the Generator, so the learning comes to an end due to slow learning (Zhang et al., 2020). Even though the plain Vanilla GAN is the simplest GAN form, still this GAN type has a few common issues while generating synthetic data in the financial domain (Sauber-Cole and Khoshgoftaar, 2022). Vanilla GAN is extremely hard to train due to multiple factors such as hyper-parameters, the loss function and the Generator, which fools the Discriminator easily. In the Vanilla GAN design, Oscillatory loss is a common issue in the training phase. An effective training process needs the loss to be stable or gradually increase/ decrease over a more extended period. On the contrary, this does not always happen in Vanilla GAN training. Another significant issue with the loss function is that it usually needs to provide more information. For instance, binary cross-entropy is the Generator's generally used loss function. One drawback of this is that there needs to be more correspondence between the output quality and the Generator's loss. Therefore, the training can sometimes be challenging to monitor (Gulrajani et al., 2017; Sabuhi et al., 2021; Takahashi et al., 2019). A categorical column imbalance is not taken into account by Vanilla GANs (McIver, 2021).

According to (Xu et al., 2019), if any rows fall into a small category, they won't be well depicted throughout training. This is a result of the random sampling of the data. They tackle this issue by resampling in such a way that discrete characteristic categories are uniformly sampled whilst still retrieving the original data distribution throughout testing. Further, generator G 1st attempts to detect the delivery within the training data before creating new data. The chance that the input data is created by noise from the training set or generator is what the discriminator has been taught to produce. The generator makes an effort to deliver data that is marginally nearer to the training sample to pretend the discriminator into thinking that the data it produces is the training dataset (Goodfellow et al., 2020). Further, (Gangwar and Ravi, 2019) employed Vanilla GANs to identify class balance in datasets. The authors pointed out that Vanilla GAN worsens the objective function of the distribution. Furthermore, it is observed that Vanilla GANs may not perform well for original datasets because they are less similar to original datasets. Moreover, (Goodfellow et al., 2014) pointed out that a min-max game's training will always be insecure. Therefore, Vanilla GANs have an issue with convergence since the moment at which to end training is unpredictable. Moreover, the mode collapse property of Vanilla GAN might result in a vanishing gradient (Goodfellow et al., 2014). Additionally, it is observed that Vanilla GAN is quite sensitive to hyperparameters (Kar et al., 2020).

The research by (Zhang et al., 2020) combined Long Short-Term Memory (LSTM) network with GAN to generate negative samples. They evaluated the validity of generated data from the distribution of the data. The empirical findings of their study show that their proposed method has a higher overall rating. In addition, the classification findings also revealed that this novel method has higher recall and precision than baseline

methods on new datasets. Although the combined LSTM and GAN model achieved excellent results, the model came across several problems while training the model. From the model outlook, discriminators and generators have to be carefully balanced. The Generator is prone to fall into the local optimum, causing a single sample and inadequate diversity. The model needs improvements from the perspective of gradient penalty, loss function and so on. Other than that, the LSTM-GAN model can only generate data similar to the original dataset and cannot generate possible patterns that have not occurred yet under the latent space. Moreover, there is a lack of diversity in the generated samples, and the model may not detect potential fraud transactions which have not occurred. Overall, the LSTM-GAN model performed very well in generating synthetic data, but the model has several loopholes which need to be addressed in future studies.

In an empirical study, (Sethia et al., 2018) employed popular adversarial networks to generate data to improve the model's performance synthetically. The study employs Least Squares, Vanilla implementation, Wasserstein, Relaxed Wasserstein and Margin Adaptive models. This study generated artificial synthetic data using different GAN variants, augmented the real dataset, and compared the results with the real dataset. Their study aimed to deal with the imbalanced class issue by generating artificial data and improving the overall training performance. The study's empirical findings (Sethia et al., 2018) found that LSGAN attained the highest performance based on classification accuracy than the other four models. At the same time, LSGAN also faced difficulties such as mode collapse as the Generator generated similar and limited data, thus leading to a loss of diversity in the produced data. Moreover, the Vanilla GAN model demonstrated considerable classification accuracy compared to the real dataset, but findings suggest that it is sensitive and unstable to its hyper-parameters. The study also indicated that the data generated by Vanilla GAN might also need more diversity and quality. Similarly, the WGAN performed better than Vanilla GAN regarding stability and classification accuracy. WGAN needs more computational resources and training time than the other models. In addition, the training phase can be complex due to the Wasserstein distance metric. Overall, all the models used in the study performed well in terms of classification performance. However, GAN-based models observed mode collapse during the training phase.

In almost all GANs, general issues may arise in the training phase via the gradient descent technique (Arjovsky et al., 2017). For instance, when the Discriminator makes a false judgement, the Generator cannot get accurate feedback, and its loss function cannot learn as it should be (Salimans et al., 2016). Similarly, if the Discriminator makes an accurate judgement, then the gradient of the loss function converges to 0. As a result, a major disturbance or delay occurs in the learning speed. Hwang and Kim (2020) argued that the WGANs had compensated for these GAN limitations as WGANs use Wasserstein distance. Wasserstein distance index uses the distance of the two probability distributions. On the other hand, the distance value is 0 under the KL Divergence when distributions overlap one another. It is constant or infinite when two distributions do not overlap each other, thus presenting an extreme distance value. The redefinition of the loss function in WGANs employing Wasserstein distance is effective as it provides smooth training and generates improved data that resemble, as much as possible, the original dataset.

In another study, (Hilal et al., 2022) argue that Vanilla GANs face the issues of vanishing gradient and mode collapse frequently. These problems are more common in real-world datasets with many modes associated with each different class. Unlike the Vanilla GANs, the loss function of WGAN depends on the quantity of generated distribution and the distance. Hence, it does not have flat regions where the distribution becomes different. As a result, the Discriminator is barred from enhanced improvement, and the challenge of vanishing gradient is addressed, while the likelihood of mode collapse challenge is reduced. The conventional GAN uses the Jensen-Shannon divergence to learn data distribution. Conventional GAN suffers from a weak, unstable signal when the Discriminator approaches a local optimum, this scenario is known as a vanishing gradient issue. The conventional GAN can also experience mode collapse due to these limitations. In addition to these issues, conventional GANs do not provide any inference model to capture the inverse mapping. Thus, additional training is required to achieve an inference model, increasing the computational cost of training (Perera et al., 2019).

To deal with the issues of conventional GANs converging to the Nash equilibrium and mode collapse, (Arjovsky et al., 2017) introduced Earth Mover distance instead of JS divergence in their study. WGAN improves GAN stability by altering the cost function. Even though the WGAN network deals with the mode collapse problem, the weight clipping employed in its Discriminator makes convergence extremely tricky. So, (Gulrajani et al., 2017) introduced an enhanced version of WGAN-GP. Researchers added a gradient penalty to the Discriminator model of WGAN as an alternative to weight clipping. This enhancement improved training speed, sample quality and convergence by allowing the Discriminator to learn smoother decision boundaries.

GANs are popular generative methods, however, they suffer from unstable training. On the other hand, Wasserstein GAN was introduced to deal with training instability, though WGAN occasionally generates poor samples or fails to converge. (Gulrajani et al., 2017) in their study highlighted that these issues are due to the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic, which can result in unwanted behaviour. This architecture employs a weight clip and adapts the evasion application's loss function to promote efficient training. They propose applying the earth mover distance to compute the loss function instead of the Jensen-Shannon divergence. This reserve metric is maintained during the whole process and is always accessible. It measures the degree to which the data distributions of the training dataset and the generated dataset are similar to one another. Further, it is observed that WGANs may not perform well for original datasets because they are less similar to original datasets. Further, it is observed that WGANs are unfortunately not as efficient as well as expected and continue to experience unsteady training and sluggish convergence following weight clipping. Gulrajani et al., (2017) have discussed the various improvements that can be done to the WGANs. They found that weight clipping, which is used to impose a Lipschitz restriction on the critic and can result in undesirable behaviour, was frequently the cause of these WGAN issues. Instead, they suggested penalising the critic's norm of the gradient concerning its input as an alternative to clipping weights (Gulrajani et al., 2017). The SDG GAN approach includes convolutional channels with an MLP architecture as both the discriminator and the generator (Xue and Zhang, 2016). A typical GAN generator attempts to create fictitious data that thoroughly resembles the real distribution (Taha and Malebary, 2020). In addition to improving the classification accuracy of standard datasets and real fraud datasets, SDG GAN has the potential to surpass density-based oversampling techniques. It is observed that SDG GAN has a lower recall value as compared to cGAN, SMOTE, ADASYN, B-SMOTE, and SMOTE (Charitou et al., 2021). However, it is observed that SDG GAN is less similar to the original datasets because this technique may not be appropriate for these datasets.

Advantages and Disadvantages of GAN-based techniques

According to recent studies (Cao et al., 2018, Mullick et al., 2019) the following are the main advantages and disadvantages of GANs:

Advantages:

- 1) GANs are highly versatile and adaptable, capable of generating data that closely resembles real data. For instance, when given an input image, GANs can generate a modified version of the image that closely resembles the original. They can also generate various versions of content, videos, and audio.
- 2) GANs excel at capturing the intricacies of data and can easily generate diverse variations, making them valuable for various AI tasks.
- 3) Unlike traditional machine learning models that require labelled data, GANs can be trained using unlabelled data as they learn the internal representations of the data.
- 4) GANs learn intricate representations of data, allowing them to tackle complex AI problems.
- 5) Compared to classical machine learning algorithms, GANs leverage adversarial training and are more effective in representation and feature learning.

Disadvantages:

- 1) GANs are more challenging to train as they require a diverse range of data to ensure accurate performance.
- 2) Generating text or speech results with GANs is complex.
- 3) Mode collapse can occur, where the generator produces limited variation in output.
- 4) Vanishing gradients can be a problem, hindering effective training.
- 5) Internal covariate shift can impact performance.

However, due to the adaptability and versatility of GANs, meticulous fine-tuning can alleviate these drawbacks, leading to optimised architecture designs that can be applied to various machine learning purposes.

2.5.5 Sampling vs GAN-based techniques

It is observed that sampling based techniques leads to overgeneralization (Chou et al., 2020). Such as, SMOTE's method is inalienably risky since it indiscriminately sums up the minority region regardless of the greater part class. This system is especially risky on account of exceptionally slanted class disseminations since, in such cases, the minority class is extremely meagre regarding the larger part class, accordingly, bringing about a more prominent possibility of class combination (Chou et al., 2020). Further, it leads to an absence of flexibility. For instance, the quantity of engineered tests created by SMOTE is fixed ahead of time, accordingly, not taking into consideration any adaptability in the re-adjusting rate. Furthermore, SMOTE are not efficient for high dimensional data (Chou et al., 2020). On the other hand, GANs are very general and adaptable, and can generate data that appears to be resembling unique data (Cao et al., 2018). In the event that input for GAN is a picture, at that point it will generate another form of the picture which will resemble the first picture. Essentially, it can generate various variants of the content, video, sound e.g. Further, GANs expand into subtleties of data and can undoubtedly decipher into various forms so it is useful in accomplishing AI work (Cao et al., 2018). Gaining marked information is a manual interaction that takes a ton of time. GANs don't require marked information; they can be prepared utilising unlabelled information as they gain proficiency with the inside portrayals of the information. As referenced earlier, GANs can learn chaotic and convoluted disseminations of information (Cao et al., 2018). This can be utilised for many AI issues. Compared to the classical machine learning algorithms, GAN operates through an adversarial training approach and is more capable in representation and feature learning (Cao et al., 2018). However, GANs models also have some disadvantages. For instance, GANs models are harder to prepare because GANs require various kinds of information to be provided to GAN consistently to check in the event that it works precisely (Mullick et al., 2019). Producing results from text or discourse is intricate. Further, GANs model has also problems of mode collapse, vanishing gradients, and internal covariate shift (Mullick et al., 2019). Therefore, the current study introduces an improved GAN variant, the K-CGAN model to address these challenges and the problem of class imbalance data.

2.5.6 Summary

After extensive review of existing techniques and literature, we have meticulously chosen GANs as the approach for our proposed model. Through rigorous research and experimentation with hyperparameters and architectures, we are presenting our innovative method K-CGAN method. This novel approach for comparison incorporates various GAN-based methods and employs cutting-edge oversampling techniques to effectively address the data imbalance problem, ensuring accurate comparisons and robust analysis.

Our K-CGAN model is purposefully designed to overcome the limitations commonly found in existing GANs. By leveraging the adaptability and versatility of GANs, our model is optimized to cater to the specific needs in credit card fraud and breast cancer data analysis. It tackles the challenges associated with GANs, effectively improving the quality and reliability of generated datasets. As a result, the performance of our classification methods is significantly enhanced, providing more accurate and reliable insights.

In summary, our K-CGAN model represents a significant advancement in the field, revolutionizing the way we address data imbalance and improve the performance of classification methods. Through a meticulous approach and attention to detail, we have ensured that our model surpasses the limitations of existing GANs, resulting in datasets of the highest quality and reliability.

2.6 Methodology

This study adopted a mixed-method research approach, combining quantitative and qualitative research methodologies to comprehensively analyse breast cancer diagnosis and credit card fraud detection. This allowed the study to leverage the strengths of both these methods and avoid expected limitations that could emerge when a single methodology is adopted. In the quantitative phase, the study utilized oversampling techniques, popular GAN methods and classification methods to identify trends and make predictions. Moreover, in the quantitative phase, the study performed a thematic analysis of both datasets to understand the implications of our findings in the broader context. Thus, the mixed-method approach allowed this study to understand the credit card fraud detection and breast cancer diagnosis phenomena and provided valuable insights into the underlying mechanism of the techniques used. Our methodology also focused on extensive feature engineering and evaluation of data augmentation and classification techniques. Therefore, we've implemented and tested multiple classification and oversampling methods to assess their performance in handling imbalanced datasets. Our study evaluates the performance of classification techniques using the K-CGAN model against baseline models. The core experiment flow is depicted in Figure 3.

In addition, the study also incorporated case studies with breast cancer and credit card datasets to provide an in-depth exploration of these instances. The first case study focused on fraudulent credit card transactions. The mixed research approach helped identify patterns of fraudulent transactions. The GAN methods, oversampling techniques, and our proposed K-CGAN model were employed to resolve the class imbalance issue. The synthetic data generated using these methods helped to improve the predictive model's performance. In the second case study, the study focused on the breast cancer diagnosis. This approach helped identify the trends and patterns within the breast cancer data. The synthetic data generated by the GANs effectively identified malignant tumours with high accuracy.

Quantitative research, an integral component of our study, involves objective data collection and systematic analysis. By adopting this approach, the study quantified relationships and made predictions. The methodology of this study began with defining research questions and hypotheses. In addition, this approach allowed us to resolve the imbalanced dataset issue by generating synthetic data for the minority class. This approach, combined with objective data collection and systematic analysis, is intended to address both datasets' imbalanced distributions, thus avoiding biased models and inaccurate classification results. For the said purpose, research questions and hypotheses were formulated to guide the collection of numerical data. The quantitative approach was adopted as it is instrumental in exploring relationships, making accurate predictions and identifying trends. In addition, common methods such as experiments, analysis and data mining techniques were used. By doing so, this study provides insights and patterns that would have otherwise remained hidden. For instance, the data mining techniques used in this study helped discover patterns in these large datasets.

In addition to quantitative research, the qualitative approach was also adopted in our research. The qualitative approach is key in any research as it provides a nuanced understanding of the data. This approach involves a more subjective, interpretative analysis to discern patterns and themes within the data. The qualitative methodology of the study began with the collection of the data. Moreover, thematic analysis, a core aspect of this study, was adopted to interpret and identify recurring themes and patterns in the data. The study carefully reviewed data, followed by coding and categorizing the data into themes. Additionally, the qualitative methodology used in this study was iterative, as repeated revisions were made to ensure the reliability and accuracy of the identified themes.

The qualitative approach also provided a deeper understanding of the imbalance class phenomena. Also, the study interpreted the reasons for the performance of various GANs and other oversampling methods based on the themes identified in the data. By doing so, the study offers insights that were difficult to achieve through the quantitative method alone. Moreover, qualitative research enables this study to comprehend the implications of our findings in the broader category of breast cancer diagnosis and credit card fraud detection. The qualitative research complemented the quantitative research of this study, thus offering an in-depth and comprehensive understanding of credit card and breast cancer datasets. The integration of both these methodologies allowed this study to provide a balanced view of our findings, thus ensuring a comprehensive and robust conclusion of our study. Our research study focuses on resolving imbalance dataset issue by generating synthetic credit card and breast cancer data for minority class transactions. The existing datasets for these tasks suffer from imbalanced distributions, which can lead to biased models and inaccurate classification results. Therefore, we aim to create a more balanced resource by exploring GANs and oversampling techniques.

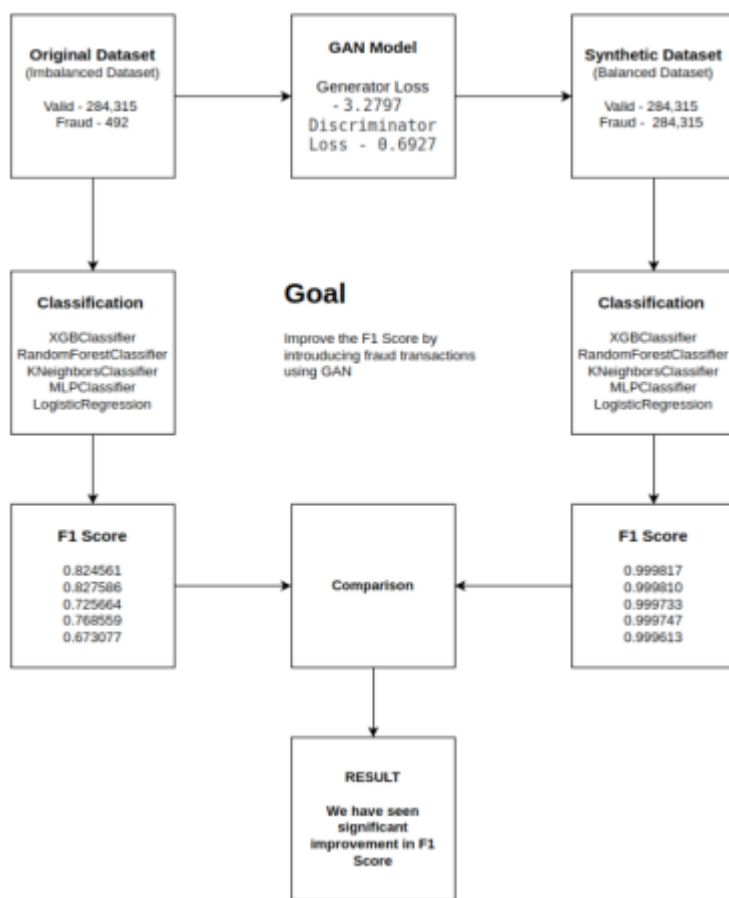


Figure 3: Experiment flow

Chapter 3

Kullback-Leibler Divergence Conditional GAN (K-CGAN)

3.1 Introduction

In this research study we have experimented GANs to generate synthetic credit card and breast cancer data for minority class transactions. Our experiments have implemented various GAN-based architectures, including WGAN (Arjovsky et al., 2017), LS GAN (Mao et al., 2017), NS GAN (Shannon et al., 2020), and SDG GAN (Charitou et al., 2021) to evaluate their effectiveness in creating high-quality synthetic data. To improve the quality of the generated data, we have developed a novel optimised custom GAN, referred to as K-CGAN. The main goal of our research is to develop a reliable synthetic dataset and address the issue of imbalanced datasets in classification tasks. The existing credit card fraud and breast cancer datasets suffer from an imbalanced distribution of data points. Therefore, our aim is to create a more balanced resource by exploring GANs and several oversampling techniques, such as ADASYN (He et al., 2008), SMOTE (Chawla et al., 2002) and B-SMOTE (Han et al., 2005), to improve our research results.

This approach aims to create a more balanced resource that can enhance the accuracy and reliability of classification models for detecting credit card fraud and breast cancer. The K-CGAN model has been extensively tested on credit card fraud and breast cancer data. The study evaluates the performance of classification techniques using the K-CGAN model against baseline models. The results demonstrate that the K-CGAN model, with its custom loss function architecture and hyperparameters, provides superior accuracy and performance compared to the baseline models, particularly in credit card fraud detection tasks. Furthermore, the K-CGAN model is capable of generating high-quality synthetic data that can be used to augment existing datasets, resulting in improved accuracy and performance. This approach offers a practical solution to address imbalanced datasets and contributes to the advancement of credit card fraud detection and breast cancer classification tasks.

3.2 K-CGAN Architecture and Implementation

Various extensions of GAN based methods have been developed since its introduction Goodfellow *et al.* (2014). To some extent, these GAN extensions have addressed the irregularities associated with the original GAN method. Motivated from the recent development in GAN-based synthetic generative frameworks we propose K-CGAN method to address the imbalanced class issue. This proposed method is based on a conditional GAN (cGAN) (Mirza & Osindero, 2014) framework with a novel custom loss function for the Generator where an additional Kullback-Leibler (KL) Divergence loss is introduced along with the binary cross entropy loss. We show in this study that addition of this KL Divergence loss helps in enhanced data sampling from latent space and smoother training convergence. Due to the addition of this custom loss

function we named our proposed method Kullback-Leibler divergence Conditional GAN (K-CGAN). Our custom K-CGAN model is displayed in Figure 4. In addition to a custom loss function, we make use of state-of-art training techniques for smoother and faster convergence e.g. Kernel Regularizers (L2), Kernel Initializers (Glorot_Uniform), Batch-Normalization, Dropouts and other optimal hyperparameters. Kernel Regularizer methods like L2 (Bishop, 2006) help in smoother training, Kernel Initializer methods like Glorot_Uniform (Glorot, 2010) help in initializing the parameters of the model (weights & biases) in suitable loss regions, helping in avoiding local optimas and faster convergence of training loss routines. The full list of hyperparameters and their values are presented in Tables 1 and 2. K-CGAN method is composed of two sub-networks: the Generator and the Discriminator. Also, the CGAN process is displayed in Figure 3.

K-CGAN comprises two Neural networks (NNs), the Generator and the Discriminator which compete with each other in adversarial settings. The goal of Generator is to learn the inherent data sampling distribution of original training samples and be able to generate more synthetic samples mimicking the original sample distribution. The goal of Discriminator is to be able to differentiate between samples from the original distribution and synthetic samples. The goal of the Generator when generating synthetic samples is to make it close enough from the original distribution so that the Discriminator is not able to differentiate it from the original distribution. Hence, the adversarial setting. Both Generator and Discriminator force each other to learn better.

Generator Network: Takes random noise and class labels as inputs and generates synthetic data samples. The generator is conditioned on the class labels, making it a conditional GAN. It consists of multiple dense layers with dropout and batch normalisation.

Discriminator Network: Takes real or generated data samples along with their corresponding class labels as inputs and classifies them as genuine or fake. The discriminator uses LeakyReLU activation, dropout, and batch normalization to discriminate between real and fake data samples.

The Generator start-off with random data distribution and attempts to imitate a specific type of distribution. Through training, the Discriminator gets better at differentiating fake distributions from the real ones. In this way, both the NNs play a mini-max game in which both try to outsmart one another. The training architecture of GAN looks simplistic, but at times, both these NNs do not learn the way they are trained to. GANs have a tendency to show some irregularities in performance. These irregularities are linked to their training and are active fields for research. Through meticulous tuning of K-CGAN we're able to achieve superior results.

Some Notation:

- x (original input sample space) εR^m
- z (latent space) εR^d
- y (class labels) εR^1 [# class labels = 2; label 1 (real samples), label 0 (fake samples)]
- \hat{x} (synthetic/fake samples generated by Generator) εR^m
- Batch_Size (b) (number of samples used for model training in each iteration)
- N (total number of training samples)
- D (Discriminator Network)
- G (Generator Network)
- $p(x), p(y)$ e.g. represent probability distributions

Generator Network (G):

- Takes
 - random sample from latent space (z_i), class label (y_i) as inputs
 - and generates a new synthetic sample (\hat{x}_i)
- The Generator output is conditioned on the class labels (y), hence the name conditional GAN

Discriminator Network (D):

- Takes
 - as input:
 - real input sample (x_i), class label (y_i)
 - or synthetic sample (\hat{x}_i), class label (y_i)
 - and tries to respectively classify them as real or fake
- The Discriminator output is also conditioned on the class labels (y), in accordance with the name conditional GAN

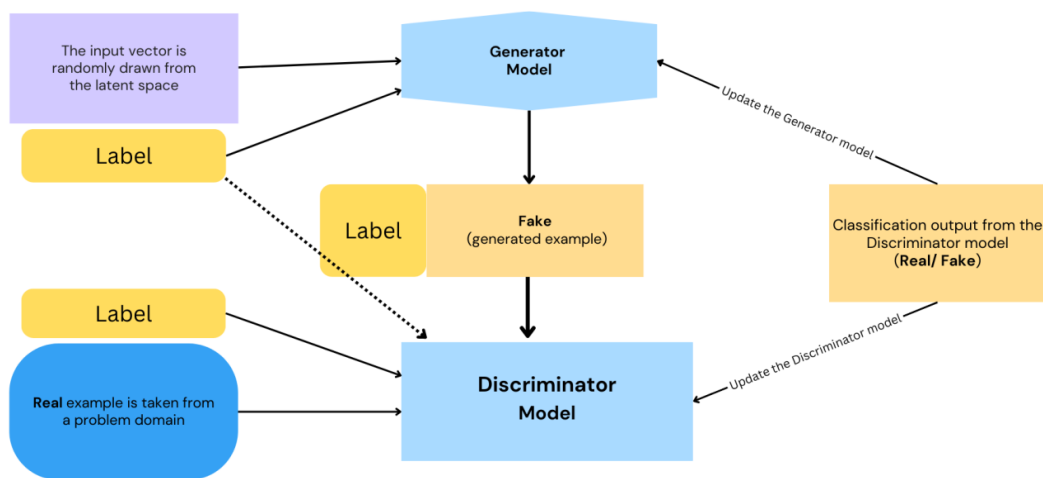


Figure 4: Process steps of CGAN architecture

Here are the key steps in the operation of CGAN architecture. A CGAN architecture uses a discriminator network to separate the real data from the created data, a generator network to create synthetic data that is reliant on the input vector, and a discriminator network to distinguish between the real and generated data (Padmanabhuni and Gera, 2022). Since the CGAN is trained by minimising the loss function, it may be used to provide synthetic data that is appropriate for a certain task or application (Wickramaratne and Mahmud, 2021). Based on the results of numerous studies (Ba, 2019; Wickramaratne and Mahmud, 2021), it is clear that GAN is a powerful tool for generating synthetic data that can be used to aid in the categorization of data, such as in credit card fraud detection. As per the above, based on CGAN architecture our proposed optimised method K-CGAN the generator G and discriminator D of K-CGAN are constantly in conflict with one another. The generator's purpose is to perplex the discriminator. The discriminator's job is to distinguish events produced by the generator from those in the provided dataset. If the discriminator has no issue identifying which instances came from the generator, the generator's data will be of a low quality. It is reasonable to think of the K-CGAN setup as the generator's training ground, with the discriminator giving the generator input on the instances it generates and guiding its evolution. The K-CGAN architecture is presented in Figure 4.

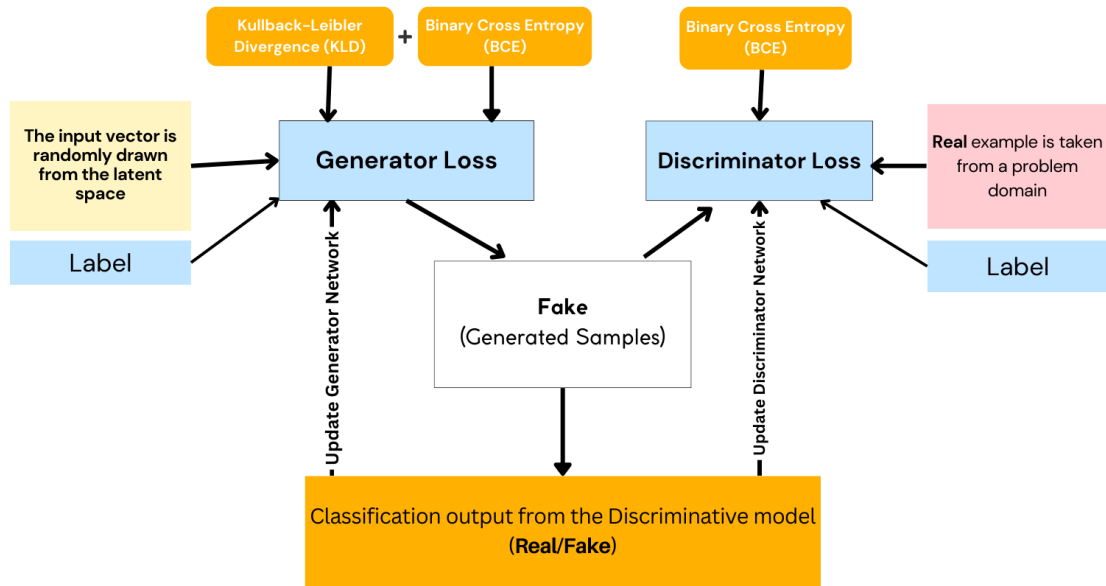


Figure 5: Architecture of K-CGAN method

During the training Generator and Discriminator are trained in sequence while freezing the parameters (weights & biases) of the other for adversarial setting to take effect. First the Discriminator parameters can be trained keeping parameters of Generator frozen and then in the next phase the Generator parameters get trained keeping parameters of Discriminator frozen.

In the initial stage of training the K-CGAN involves generating synthetic samples from random noise vectors of latent space and class label space. Generator takes input of dimensionality $[\mathbf{b}, 2\mathbf{d}]$. Since we are working with CGAN we incorporate class label ($y_i \in R^1$) after passing it through an *Embedding Layer* and concatenating it to the random noise sample ($z_i \in R^d$) of latent space. The Generator of novelty K-CGAN transforms this batch of $[\mathbf{b}, 2\mathbf{d}]$ input samples into output batch of synthetic data of dimensionality $[\mathbf{b}, \mathbf{m}]$, as each output sample (\hat{x}_i) is of same dimensionality as input (x_i). The Discriminator links both the real data and the generated synthetic data with their labels to shape the combined labels and combined data. Our proposed framework then prepares target labels to form a binary label that discriminates real data from fake data. Layer transformations and input-output dimensionality are explained clearly in Figure 5.

The next phase of K-CGAN involves training the Discriminator. In this phase, the Discriminator is used to classify the combined labels and data to minimise the Discriminator loss, defined by the loss function between the predictions and true labels. During this training phase parameters of Generator remain frozen and parameters of Discriminator get trained. The gradients of the loss with respect to the Discriminator's trainable weights were computed and used to update the parameters of the Discriminator. The Discriminator's loss was calculated in this stage of training. The Discriminator loss is the binary cross-entropy between misleading labels and prediction. Furthermore, the gradients and update weights were calculated. This process is known as back-propagation, in which the gradients of the loss with respect to the weights are calculated. The optimizer was then utilised to update the weights, ultimately leading to the minimum possible loss.

The next steps of training novelty K-CGAN involve the preparation of data and the start of the Generator training. In this training phase, the Generator is trained to transform the noise and real labels into fake data, and uses the Discriminator to classify these fake data samples. Moreover, the Generator aims to maximise the loss defined by the loss function between the misleading labels and the Discriminator's predictions of the fake data. The Generator loss was calculated after the Discriminator loss. The Generator loss consists of two terms: binary cross-entropy between true labels and predicted labels by the Discriminator, and KL Divergence between the original data and fake data generated by the Generator. The Kullback-Leibler (KL) divergence is a measure of the difference between two probability distributions. This function is used to optimise the parameters of the Discriminator model based on the difference between the original data distribution and the fake data distribution of the Generator model. In the last step of training, tracking of the Discriminator and the Generator loss is performed using the Generator loss tracker and Discriminator loss tracker objects, returning these losses as a dictionary.

KL Divergence is a way to measure matching between two distributions, and in our case, the Original and Generated Data are our two distributions. As we consider KL Divergence as a loss function our goal is to have minimum loss and as the training process goes, both distributions come close to each other, as demonstrated in Figure 5. In most of the GAN networks, difference between two distributions is never considered as a loss. The main loss function is binary classification between real and fake samples. So as we added KL Divergence in the Generator loss and if KL loss approaches 0 (zero), it means both distributions are achieving optimum closeness.

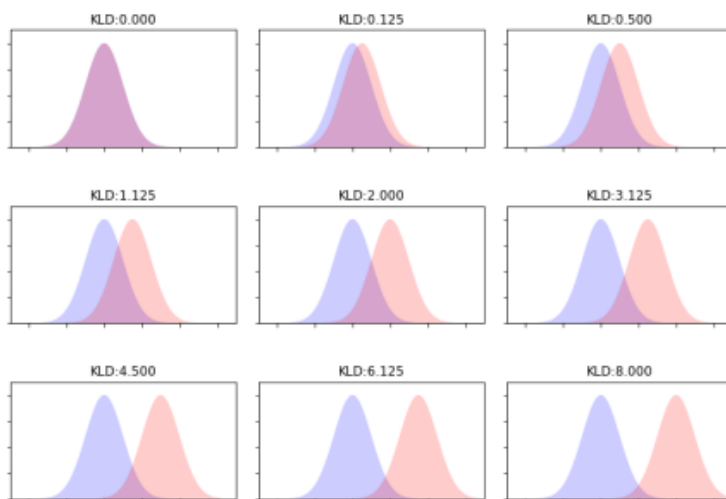


Figure 6: For similar probability distributions KL Divergence is closer to 0 and for dissimilar probability distributions they are high values highlighting divergence

To create convincing synthetic fraudulent transactions, a novel loss approach has been suggested for the generator to guarantee that an accurate classification model can be effectively trained. In order to ensure that the K-CGAN-generated synthetic data properly reflects the distribution of the original dataset, we utilised KL Divergence to guarantee higher accuracy performance. Since the KL divergence is continuously differentiable, gradient-based optimisation methods such as deep learning may utilise it as a loss function (Günder et al., 2022). In the field of information theory, the KL divergence, also known as relative entropy, was first presented by Solomon Kullback and Richard Leibler in 1951. By comparing two probability distributions, this statistical tool calculates the distance between them (Chen et al., 2018). Further, Xu and Veeramachaneni (2018) used TGAN model to detect frauds in credit card transactions. They used Adam Optimizer to train their model (LeCun et al., 2015). They optimised the generator to make it as effective at deceiving the discriminator as feasible. They jointly optimise the cluster vector of continuous variables and the KL divergence of discrete variables by adding them to the loss function in order to more effectively warm

up the model. The KL divergence term can help improve the stability of the model (Xu and Veeramachaneni, 2018). To efficiently produce discrete features, they modify the loss function by including noise and KL divergence. They observe that GANs are more scalable for big datasets and can efficiently capture the correlations between features.

However, it is observed that the proposed K-CGAN model provides better parameters performance by using KL divergence because it uses combined binary cross entropy and KL divergence in generator loss while still using binary cross entropy in discriminator loss, along with the optimised custom hyperparameter settings and layers for each dataset. This ensures that our synthetic data is sufficiently precise and accurately resembles its source material. By utilising this technique for credit card fraud detection, financial institutions can trust in more dependable results from our project. Finally, to test the efficacy of K-CGAN-generated synthetic data in credit card fraud detection, K-CGAN model was trained on a credit card fraud transaction dataset and its performance has been evaluated using metrics such as precision, recall, and F1 score. Further analysis, training and optimization of the K-CGAN on breast cancer data has revealed that utilising these synthesised datasets improves the accuracy of minority class detection models significantly. Additionally, other GAN-based and oversampling techniques were trained using the same datasets as K-CGAN and assessed for their performance.

K-CGAN implementation

The K-CGAN training procedure is remarkably similar to that of the Conditional GAN. The logistic cost function for the gradient is obtained by feeding a mini batch (b) of training samples and noise random samples. In order to trick the Discriminator into categorising the dataset and create it as the training dataset, the Generator seeks to provide data that are relatively close to the training set. The Generator is trained to generate fake data samples that are conditioned on certain additional information, such as class labels. The Generator model is designed to take two inputs:

'Noise Input' is a random noise vector sampled from a normal distribution in the latent space (R^d). It acts as a source of learning training data sample distribution for the Generator and helps generate diverse new synthetic/fake data samples which are close to the original data sample distribution.

'Label Input' is the class label corresponding to the data samples that the Generator needs to generate. For example, in our dataset, the label "0" corresponds to fraudulent Finance transactions, and the label "1" corresponds to valid transactions.

The idea is to guide the Generator in producing data samples that are specifically tailored to a particular class. By providing the label as an additional input to the Generator, the model can generate data samples that align with the desired class and hence the name Conditional GAN for this kind of method. The noise input and the transformed label input are then concatenated to create a combined input for the Generator. This combined input is fed into a series of dense layers to generate the fake data samples. The output of the Generator is passed through a tanh (Hyperbolic Tangent) activation function to ensure that the generated data is within the range of -1 to 1, which is the desired range of the dataset after normalisation. By providing both the noise input and the label input to the Generator during training, the Generator learns to generate data samples that align with specific classes, as specified by the label input. This way, the Generator can produce realistic fake data samples that resemble either valid or fraudulent transactions based on the given label.

Training and Testing

Table 1 highlights the custom optimised hyperparameter values for the training process of the K-CGAN for credit card fraud data. The goal of these hyperparameters was to ensure effective training while mitigating the impact of noise on the model's performance. To address the influence of noise, the activation function of

the nodes in the K-CGAN method was deliberately set to minimize its impact. Previous studies commonly employed the Rectified Linear Unit (ReLU) and leaky ReLU activation functions due to their superior convergence properties and faster computation. In this study, the Generator neural network leveraged the widely-used ReLU activation function, while the Discriminator neural network incorporated the LeakyReLU activation function known for its exceptional performance on complex datasets.

The loss function of Generator network is a modified version of binary cross entropy loss consisting only of fake data (as during Generator network training we are only working with fake samples), along with an additional KL Divergence loss which we introduce in our study. This hybrid custom loss function consisting of modified binary cross entropy and KL Divergence loss for Generator network ensures that the generated samples closely resemble the original data distribution. The exact derivations of Discriminator loss function are given in Eq. (24 – 28). The exact derivations of Generator loss function is given in Eq. (36 – 37). Both the Generator and Discriminator networks consisted of three layers, each with specific neuron sizes. The Generator network has three layers (2 hidden layers, 1 output layer) with 64 , 32, 29 neurons resp., the Discriminator network has three layers (2 hidden, 1 output layer) with 20, 15 and 1 neurons resp. Network and layer architecture are clearly explained in Figure 6 (c). To prevent overfitting and enhance generalization performance, a dropout value of 0.1 was applied to both networks.

For optimization, the Adam optimizer was used for training both the Generator and Discriminator networks. The Adam optimizer is known for its excellent convergence rate and stability, making it an ideal choice for this task. A learning rate of 0.0001 was adopted for both the Generator and Discriminator to ensure convergence to the global minimum. Moreover, the Generator network employed a random noise vector size of 100 (d) for generating synthetic data. The kernel initializer utilized in the Generator network was `glorot_uniform`, which uniformly initializes weights and effectively addresses the exploding gradient problem. Additionally, a kernel regularizer based on L2 regularization was implemented in both the Generator and Discriminator networks to control overfitting during the training process. These hyperparameter choices were carefully selected and fine-tuned to achieve optimal performance in generating synthetic data that closely resembles the real credit card fraud data distribution while minimizing the impact of noise and overfitting.

Table 1: Novelty K-CGAN Optimised hyperparameter settings for credit card fraud data

Hyperparameter	Generator Neural Network	Discriminator Neural Network
Activation	ReLU	LeakyReLU
Loss function	Modified Binary Cross Entropy + KL Divergence	Binary Cross Entropy
Hidden Layers	(3 - 2 hidden, 1 output) 64, 32, 29	(3 - 2 hidden, 1 output) 20, 15, 1
Dropout	0.1	0.1
Output Optimizer	Adam	Adam
Learning Rate	0.0001	0.0001
Random Noise Vector	100	-
Kernel Initializer	glorot_uniform	-
Kernel Regularizer	L2 method	L2 method
Total Learning Parameters	36,837	1,519

Further, Table 2 presents the custom optimised hyperparameter settings for the K-CGAN model for the breast cancer data. The table presents the hyperparameters for the generator and the discriminator neural networks. The Generator network has three layers (2 hidden layers, 1 output layer) with 64, 32, 29 neurons respectively, the Discriminator network has three layers (2 hidden, 1 output layer) with 20, 15 and 1 neurons respectively. The activation function used in the generator neural network is ReLU. The loss function used in the generator neural network is the trained discriminator loss plus KL divergence. The output optimizer used in the generator neural network is Adam, and the learning rate is set to 0.0001. The generator neural network also includes a dropout layer with a dropout rate of 0.2. The activation function used in the discriminator neural network is LeakyReLU. The loss function used in the discriminator neural network is binary cross-entropy. The output optimizer used in the discriminator neural network is Adam, and the learning rate is set to 0.0001. The discriminator neural network also includes a dropout layer with a dropout rate of 0.2.

Table 2: Novelty K-CGAN Optimised hyperparameter settings for breast cancer data

Hyperparameter	Generator Neural Network	Discriminator Neural Network
Activation	ReLU	LeakyReLU
Loss function	Modified Binary Cross Entropy + KL Divergence	Binary Cross Entropy
Hidden Layers	(3 - 2 hidden, 1 output) 64, 32, 29	(3 - 2 hidden, 1 output) 20, 15, 1
Dropout	0.2	0.2
Output Optimizer	Adam	Adam
Learning Rate	0.0001	0.0001
Random Noise Vector	100	-
Kernel Initializer	glorot_uniform	-
Kernel Regularizer	L2 method	L2 method
Total Learning Parameters	10,226	1,386

3.3 The Application of Kullback-Leibler (KL) divergence in GANs

In probability theory, statistics, and information theory, the Kullback-Leibler divergence (KL-divergence) is a well-known quantity. It was first proposed by Kullback and Leibler (Kullback and Leibler, 1951) and measures relative entropy in the domain of the theory of information simultaneously evaluating the similarity among two distributions of probability in the framework of statistics and probability. Further, an indicator of how divergent two probability distributions are from one another is the KL divergence. Furthermore, KL-divergence has several uses, including multivariate data analysis, estimation, approximation, and regression. Examples of these applications include pattern recognition and discriminant evaluation (Olszewski, 2012a). In other words, KL divergence is a metric in machine learning and information theory to compare the similarity between two probability distributions (Doria et al., 2022; Pitsane et al., 2022). It measures how different two probability distributions are from each other and can be interpreted as a measure of the bits of information lost when using one distribution as an approximation for the other. It is often used in machine learning to compare a model's predicted distribution to the true distribution of the data (Langevin et al., 2022).

Different scholars have used KL divergence in GANs model for different purposes. For instance, Olszewski (2012) employed the Latent Dirichlet Allocation method of GANs model to detect fraud in telecommunications. The authors pointed out that KL divergence in GANs is an indicator of how divergent two probability distributions are from one another. The KL-divergence is a key component of the proposed method for fraud detection in telecommunications (Olszewski, 2012a). It is used to evaluate the difference between a classified account's LDA model and a reference account's LDA model. This evaluation is then used to classify the account as either fraudulent or non-fraudulent. Furthermore, the KL-divergence is a metric for determining the distinction among two distributions of probability (Breskuviene and Dzemyda, 2023). This study, it is used to compare the LDA models of a classified account and a reference account. The

LDA model represents the topics that are most likely to be associated with the account's communication patterns (Olszewski, 2012a). By comparing the LDA models using KL-divergence, the method can determine whether an account is more similar to fraudulent or non-fraudulent accounts. Further, the paper proposed 3 techniques for resembling the KL divergence among LDAs: Monte Carlo sampling, variational inference, and Taylor series expansion. These methods were evaluated in an experimental study which showed that Monte Carlo sampling provides the most accurate results.

Furthermore, the use of KL-divergence in this study allows for a probabilistic approach to fraud detection in telecommunications that takes into account the unique communication patterns of each user (Park et al., 2021). Moreover, the authors argued that the Kullback-Leibler (KL) divergence is a key component of the proposed method for fraud detection in telecommunications. It is used to evaluate the difference between a classified account's LDA model and a reference account's LDA model. This evaluation is then used to classify the account as either fraudulent or non-fraudulent. Furthermore, Langevin et al., (2022) used the GANs model to detect fraud and augmented data in credit card transaction data. In the context of their study, KL Divergence is used to compare the observed error rates in a model to the true error rates (Langevin et al., 2022). This means that the difference between the observed and true error rates is very small, with a high degree of confidence (Langevin et al., 2022).

Moreover, Olszewski (2012b) employed Kullback-Leibler divergence and Latent Dirichlet Allocation for fraud detection in telecommunications. Dominik Olszewski provided a detailed discussion of KL-divergence and its application in fraud detection. In the context of their study it is used to compare the probability distributions of topics generated by LDA for different telecommunications accounts (Olszewski, 2012b). Further, KL-divergence is a useful tool for comparing probability distributions and detecting anomalies in data. The paper provides a thorough explanation of its application in fraud detection using LDA. In addition to its use in fraud detection, as discussed in the previous answer, KL-divergence has many other applications. For example, in machine learning, KL-divergence can be used as a function for training generative models such as variational autoencoders. Further, in information theory, KL-divergence can be used to quantify the amount of information gained when moving from one probability distribution to another. However, in statistics, KL-divergence can be used to compare different models or estimate model parameters.

Furthermore, Zioviris et al., (2022) employed a deep learning multistage model to detect frauds in credit card transactions. In the context of this article on credit card fraud detection using a deep learning multistage model, KL divergence is used as a loss function to train the deep learning models. Specifically, the authors use a multistage model consisting of three stages: feature extraction, anomaly detection, and classification (Zioviris et al., 2022). The anomaly detection stage uses KL divergence as its function to measure the difference between the probability distribution of normal transactions and that of anomalous transactions. In the context of anomaly detection, KL divergence can be used to identify the difference between the possibility of dissemination of normal transactions and that of anomalous transactions. Further, the goal of this study is to identify anomalous transactions that deviate significantly from normal behaviour. To do this, the model first learns a representation of each transaction using a deep neural network in the feature extraction stage. Then, in the anomaly detection stage, it uses KL divergence as a measure of how different each transaction's representation is from that of normal transactions (Zioviris et al., 2022). Moreover, when comparing the performance of both models, K-CGAN outperformed the others. In particular, K-CGAN demonstrated a precision of 0.999598, recall of 0.9972, accuracy of 0.9984, and F1-score of 0.998400 which will be further presented in the upcoming chapter.

Jiang et al., (2018) used KL divergence as one of the metrics to evaluate their proposed method for fraud detection. Specifically, they calculate the KL divergence among the dissemination of features in normal transactions and that in fraudulent transactions. They then use this metric to determine whether a new transaction is more likely to be normal or fraudulent. To calculate KL divergence, we first need two probability distributions: P and Q. These distributions can represent anything from the frequency of certain

words in a text corpus to the likelihood of different types of transactions occurring in a credit card dataset (Jiang et al., 2018). Additionally, the authors present the performance measures for their proposed method, AggRF+FB (aggregation strategy with a feedback mechanism), using various feature selection techniques and classifiers. The precision values range from 0.965 to 0.985, recall values range from 0.935 to 0.975, F1-score values range from 0.950 to 0.980, and accuracy values range from 0.970 to 0.990. However, upon comparing the performance of both models, K-CGAN demonstrates superior results. In comparison, K-CGAN achieved a precision of 0.999598, recall of 0.9972, accuracy of 0.9984, and F1-score of 0.998400. Once researchers have these distributions, they can calculate KL divergence. In credit card fraud detection, researchers might use KL divergence to compare the distribution of features (such as transaction amount or location) in normal transactions to that in fraudulent transactions. Further, they could then use this metric to determine whether a new transaction is more likely to be normal or anomalous. For example, Bockel-Rickermann *et al.* (2022) planned an HMM-based method for detecting fraud in real time for merchants. They used KL divergence as one of their metrics for detecting concept drift (i.e., changes in the underlying distribution of data).

Moreover, the authors compare the accuracy and recall of their proposed method using AggRF+FB with those of two other methods (AggRF and RawLR) (Chole et al., 2022). The outcomes display that their suggested technique attains higher accuracy and recall than the other two methods (Jiang et al., 2018). Overall, these results demonstrate that the proposed method using AggRF+FB with appropriate feature selection techniques and classifiers can effectively detect credit card fraud with high precision, recall, F1-score, and accuracy values compared to existing methods (Chole et al., 2022). To evaluate their proposed method, the authors use several metrics, comprising recall, precision, F1-score, and KL divergence. They calculate KL divergence among the dissemination of features in normal transactions and that in fraudulent transactions. They then use this metric to determine whether a new transaction is more likely to be normal or anomalous. Moreover, the authors pointed out that KL divergence is used as a metric to appraise the effectiveness of the proposed method for credit card fraud recognition and as a tool for detecting concept drift.

On the other hand, Shen (2021) argued that KL Divergence is a commonly used measure in machine learning and information theory, and it has been used in various fraud detection models. It is often used to compare a model's anticipated probability distribution with the actual probability distribution of the data (Shen, 2021). In credit card fraud detection models, KL Divergence can be used to measure the difference between the normal transaction distribution and the fraudulent transaction distribution. For example, Lei and Ghorbani (2012) used KL Divergence to compare the normal network traffic distribution with the anomalous network traffic distribution in order to detect network intrusions. However, anomalies are transactions that deviate significantly from the normal transaction pattern (Lei and Ghorbani, 2012). By comparing the normal transaction distribution with the fraudulent transaction distribution using KL Divergence. Moreover, in the context of credit card fraud detection using autoencoder-based deep neural networks (Shen, 2021). The autoencoder would be trained to diminish the KL Divergence among the input data and its reassembled output. This would encourage the autoencoder to absorb a compacted depiction of the input data that captures its essential features while filtering out noise and irrelevant information. Further, the authors report the accuracy value as the primary performance measure for their hybrid model. The accuracy values for the hybrid model has an accuracy value between 0.9561 and 0.9637, with a mean AUC value of 0.9609 and a standard deviation of 0.0022. The individual models for normal transactions and fraud transactions have accuracy values between 0.9525 and 0.9543, with mean AUC values of 0.9537 and 0.9534, respectively. However, when the performance of both the models compared then the results of K-CGAN displays improved performance because in comparison, K-CGAN achieved accuracy of 0.9984. Further, the article does not report other performance measures such as precision, recall, F1 score, it does mention that the hybrid model has a mean accuracy value.

Moreover, Anh et al., (2020) used KL divergence to estimate the marginal log-likelihood of input data, which is a measure of how well their model fits the data. It is often used in machine learning and information theory to compare an approximate distribution to a true distribution (Anh et al., 2020). By using KL divergence in this way, they can compare their approximate posterior distribution to the true posterior distribution and estimate how well their model fits the data (Anh et al., 2020). This allows them to detect fraud in credit card and e-commerce transactions more accurately. Their suggested technique achieved a precision of 97.62%, which is higher than the values reported by (Jain et al., 2016) and (Srivastava et al., 2008). The anticipated technique yielded impressive results, with a recall of 97.94%, an F1-score of 96.69%, and an accuracy of 97.33%. Upon comparing the performance of both models, it became evident that K-CGAN outperformed the deep neural variational autoencoder oblique random forest. The latter achieved a precision of 97.62%, a recall of 97.94%, an accuracy of 97.33%, and an F1-measure of 96.69%. In contrast, K-CGAN exhibited superior performance, boasting a precision of 99.98%, a recall of 99.72%, an accuracy of 99.84%, and an F1-score of 99.84%.

Moreover, KL divergence plays a key role in this article's proposed method by providing a loss function for training the generator network in their semi-supervised GAN framework. In this article, the authors evaluate the performance of their proposed semi-supervised GAN framework for fraud detection in online gambling using several performance measures (Charitou et al., 2020). These measures include accuracy, recall, precision, and F1-score. The authors present the mean values and standard deviations for performance measures from 10 different runs in their experiments. They report an average accuracy of 0.998, an average recall of 0.999, an average precision of 0.997, and an average F1-score of 0.998. When comparing the performance of both models, K-CGAN demonstrates superior results. In comparison, SSGAN achieved a precision of 99.54%, recall of 86.73%, accuracy of 97.07%, and F-measure of 92.31%. On the other hand, K-CGAN achieved a precision of 99.98%, recall of 99.72%, accuracy of 99.84%, and F1-score of 99.84%. These impressive values indicate that the K-CGAN method excels in detecting fraud in credit card data. However, it is important to note that these results are specific to their dataset and experimental setup, and further testing and validation are necessary to determine their generalizability to other datasets or applications.

3.2.1 KL-Divergence Loss Function

In order to ensure that the GAN-generated synthetic data properly reflects the distribution of the original dataset, the K-CGAN method utilised **KL Divergence** to guarantee high accuracy according to Shlens (2014). Furthermore, KL divergence is a commonly used metric in machine learning and information theory to compare the similarity between two probability distributions (Doria et al., 2022; Pitsane et al., 2022). It is often used in machine learning to compare a model's predicted distribution to the true distribution of the data (Langevin et al., 2022). KL divergence is a primary equation of machine learning that measures the nearness of two probability distributions. In another study, Weijs et al., (2010) argued that KL divergence in statistics that measures the closeness of probability distribution P to a model distribution Q.

The KL divergence or relative entropy is a way to measure the dissimilarity between two probability distributions (Luo et al., 2022). The KL divergence between two probability distributions, P and Q is usually represented using the below term (Brownlee, 2019).

KL is a non-systematic metric that measures the difference in information or relative entropy characterized by two distributions. In other words, it measures the distance between two data distributions signifying how the distributions vary from one another. These probability distributions are $P(x)$ and $Q(x)$. The idea of the KL divergence score is that if the probability of an event in P is larger and the probability for that event in Q is smaller, then the divergence will be larger. On the other hand, if the divergence from

P is smaller than the probability from Q, the divergence will be larger too but not as that in the first case. The KL divergence between two distributions P and Q is usually represented using the notation below:

$$KL(P||Q) \quad (18)$$

Where, “||” point to the divergence of P from Q.

Moreover, in order to calculate the KL Divergence, there are two forms: discrete form and continuous form. The discrete form of KL divergence is represented as follows:

$$D_{KL}(p(x)||q(x)) = \frac{1}{N} \sum_{i=1}^N [p(x_i) \log(\frac{p(x_i)}{q(x_i)})] \quad (19)$$

Where " D_{KL} " is " ≥ 0 " and non-symmetric in p and q . Moreover, x_i represent input data samples. While, $p(x_1), \dots, p(x_N)$ and $q(x_1), \dots, q(x_N)$ represent probability distribution values for the observed data samples and $D_{KL}(p(x)||q(x))$ calculates the difference in probability values to measure how much distinct p and q are, for instance when p and q are closer to each other as probability distribution, they would assign similar values to each x_i and " D_{KL} " would be 0 but when p and q are very distinct probability distributions " D_{KL} " would be non-zero and higher the magnitude of " D_{KL} " more the divergence a.k.a. dissimilarity between p and q . More details about the KL Divergence can be found in literature e.g. Taboga, Marco (2021). Moreover, KLD becomes zero if these distributions are the same and can more likely be close to infinity. Moreover, a common practical interpretation of KLD is that it is the “coding penalty” related to choosing distribution q to approximate the true distribution of p (Weijs et al., 2010).

Moreover, there is a continuous form of the KL divergence which is represented as below:

$$D_{KL}(p(x)||q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} . dx \quad (20)$$

The KL divergence is utilised to minimise the difference between the distribution of the synthetic data created by the GAN and the distribution of the original data in the context of the suggested novelty loss strategy for the generator in the credit card fraud detection. The K-CGAN is urged to produce synthetic data that closely matches the properties of the original data by minimising the KL divergence between these two distributions.

The use of KL divergence (KLD) in the loss function of a K-CGAN has various advantages:

- a) ***Encourages the generator to produce samples that are similar to the real data:***
The generator is encouraged to produce samples that closely match the distribution of the real data by minimising the KLD between the distribution of the generated data and the distribution of the real data. This results in higher-quality synthetic data and better performance of downstream tasks such as classification.
- b) ***Helps prevent mode collapse:*** Mode collapse is a typical problem in GANs, when the generator creates just a few samples that are highly similar to each other, resulting in a confined diversity of produced samples. By adding KLD into the loss function, the generator is encouraged to generate a broader variety of samples that encompass the complete true data distribution.
- c) ***Provides a more stable training process:*** The inclusion of KLD in the loss function can enhance K-CGAN training stability by preventing the generator from diverging and creating low-quality samples. This is due to the fact that KLD is a more stable metric than other distance measurements such as the Euclidean distance or the Wasserstein distance.

- d) **Allows for better control of the generated data distribution:** The user may regulate the degree of similarity between the produced and real data distribution by altering the weight of the KLD component in the loss function. This provides greater flexibility and control over the generated data distribution and can help to achieve better performance in downstream tasks.

In general, including KLD into the K-CGAN loss function has proven to increase the quality and variety of generated samples, allow greater control over generated data distribution, and result in a more stable and reliable K-CGAN training.

3.2.2 The Binary Cross-Entropy Loss

Binary cross-entropy (BCE) loss function, also known as log loss (Galdran et al., 2023), is a widely used loss function in machine learning, particularly in binary classification problems. The BCE is a model metric that locates incorrect labelling of the class of data by a model (Saxena, 2021). It calculates the distance between the predicted probability distribution and the true probability distribution by computing the average difference between the presented output and the ideal output (Saxena, 2021).

The low log value indicates higher accuracy value. The BCE loss function is widely used in the training of NNs.

The mathematical representation of BCE loss function is given below:

$$BCE_Loss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (21)$$

Where x_i are training data samples, y_i stand for the actual class labels and the probability of the actual class y_i predicted by the model is represented as $p(y_i)$ or \hat{y}_i . Moreover, the probability of the class one is $p(y_i)$ or \hat{y}_i and the probability of class zero is " $1 - p(y_i)$ " or " $1 - \hat{y}_i$ " (Saxena, 2021). For smoother training purposes in machine learning and deep learning we work with *log probabilities* e.g. $\log(\hat{y}_i)$ because for low probability value \hat{y}_i , $\log(\hat{y}_i)$ is a large negative number and hence leads to smoother training for gradient updates. Total number of data samples is $N[i \in N]$. Binary cross-entropy is widely used in the loss function of a GAN a neural network architecture used for creating synthetic data that is similar to real data. In GANs, two networks are trained together - a generator network that creates synthetic data and a discriminator network that distinguishes between the synthetic and real data. The goal of the K-CGAN is to find the optimal balance between the generator and the discriminator network, such that the generator creates synthetic data that is convincing and the discriminator is unable to distinguish between the synthetic and real data.

The binary cross-entropy loss function is used in K-CGAN to guide the learning of the discriminator network. The discriminator is trained to maximise the binary cross-entropy loss, penalizing incorrect predictions and improving its ability to distinguish between synthetic and real data. On the other hand, the generator is trained to minimise the binary cross-entropy loss of the discriminator, producing synthetic data that is more similar to real data.

Binary cross-entropy offers several advantages in K-CGAN:

- a) **Well-Defined and Widely Used Loss Function:** Binary cross-entropy is one of the most well-defined loss function in deep learning applications. The mathematical interpretation of binary cross-entropy is straightforward and easy to grasp. Its primary function is to measure the dissimilarity between two probability distributions - the predicted and the actual distribution of the target variable. By minimising the loss function (binary cross-entropy) using gradient descent techniques, the GAN architecture can be easily optimised. This optimization process enables the creation of high-quality synthetic data, which can be used to train deep learning models for a wide range of applications in image and text processing.

- b) **Robust to Class Imbalance:** Class imbalance is a common issue in real-world datasets, where data distributions may be skewed towards one class more than the other. This can affect the performance of a GAN architecture that relies on a loss function that is not robust to class imbalance. However, binary cross-entropy is known to be robust to class imbalance, making it an ideal choice for GAN applications. This is because the loss function's computation does not care about the frequencies of the target variable classes but rather the probabilities assigned to them. By using binary cross-entropy as a loss function in K-CGAN, it becomes more resilient to class imbalance and more capable of producing quality synthetic data.
- c) **Creation of Diverse and Realistic Synthetic Data:** One of the most significant advantages of binary cross-entropy in GANs is its capability to produce realistic and diverse synthetic data. An important aspect of GANs is their ability to learn the underlying distribution of the real data, which they then use to generate new data points. The use of binary cross-entropy in GANs allows the generator network to learn a distribution that is as close as possible to the original data distribution. This results in the generation of diverse and realistic synthetic data, which can closely mimic the real data. The produced synthetic data can be used in various applications, including training deep learning models, data augmentation, and vision-based tasks like image recognition and segmentation.

3.2.3 Loss Functions of K-CGAN

A loss function quantifies how well a machine learning model is performing on a given task or issue. A machine learning algorithm's objective is to minimise the loss function, which implies it is looking for the optimum combination of parameters to use in order to complete the task at hand. In GAN models Discriminator (D) and Generator (G) play a two-play minimax game in adversarial settings for value function $V(G,D)$. In this setting both Generator and Discriminator force each other to learn better. Generator becomes better at generating synthetic fake data close to that from original training sample distribution. The Discriminator learns to differentiate between samples from the original distribution and synthetic samples. The goal of the Generator when generating synthetic samples is to make it close enough from the original distribution so that the Discriminator is not able to differentiate it from the original distribution. Hence, the adversarial setting. The Discriminator tries to maximise the value function $V(G,D)$ while Generator minimises the value function $V(G,D)$.

$$\min_G \max_D V(G,D) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (22)$$

which can be equivalently represented as:

$$\min_G \max_D V(G,D) = \frac{1}{N} \sum_{i=1}^N [\log(D(x_i)) + \log(1 - D(G(z_i)))] \quad (23)$$

Traditional Discrimination Training:

$$\text{maximise} \left[\frac{1}{N} \sum_{i=1}^N \log(D(x_i)) + \log(1 - D(G(z_i))) \right] \quad (24)$$

which for binary classification cases (2 labels) can be equivalently written as:

$$\text{maximise} \left[\frac{1}{N} \sum_{i=1}^N [y_i \log(D(x_i)) + (1 - y_i) \log(1 - D(G(z_i)))] \right] \quad (25)$$

or as in machine learning convention of using \hat{y}_i as model outputs/predictions:

$$\text{maximise} \left[\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \right] \quad (26)$$

or equivalently as minimising the *BCE_Loss*:

$$\text{minimise} \left[-\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \right] \quad (27)$$

e.g.

$$\text{Traditional } D_Loss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (28)$$

Traditional Generator Training:

$$\text{minimise} \left[\frac{1}{N} \sum_{i=1}^N [\log(D(x_i)) + \log(1 - D(G(z_i)))] \right] \quad (29)$$

which for binary classification cases (2 labels) can be equivalently written as:

$$\text{minimise} \left[\frac{1}{N} \sum_{i=1}^N [y_i \log(D(x_i)) + (1 - y_i) \log(1 - D(G(z_i)))] \right] \quad (30)$$

but, since during training of Generator we are only dealing with fake inputs e.g. $y_i = 0$ it becomes:

$$\text{minimise} \left[\frac{1}{N} \sum_{i=1}^N \log(1 - D(G(z_i))) \right] \quad (31)$$

or as in machine learning convention of using \hat{y}_i as model outputs/predictions:

$$\text{minimise} \left[\frac{1}{N} \sum_{i=1}^N \log(1 - \hat{y}_i) \right] \quad (32)$$

which is modified version of *BCE_Loss* with first term as zero e.g.:

$$\text{Traditional } G_Loss = \frac{1}{N} \sum_{i=1}^N \log(1 - \hat{y}_i) \quad (33)$$

Novel K-CGAN Generator Loss:

In this study, along with using traditional Discriminator and Generator losses, we have additionally added KL divergence loss to our Generator loss to ensure better sampling of the generated fake data distribution. The main goal of a custom generator loss is to encourage the Generator to produce samples that are qualitatively close to the distribution of the training data, in order to generate new instances that can expand the training set and improve the generalisation of the model.

$$KLDivergenceLoss = D_{KL}(\text{Sample Data}, \text{Original Data}) \quad (34)$$

KL divergence is a measure of the difference between two probability distributions, and can be used to compare the distribution of real and fake samples in terms of their feature distributions, rather than just their binary labels. By adding KL divergence loss to the Generator, we can encourage it to produce samples closer to that of original data distributions. Adding **KL Divergence Loss** to the Generator increases the computational cost and training time of the model, as it requires calculating the distributions of the real and fake samples and computing their divergence. Therefore, it was necessary to tune the weighting of the different loss terms and the learning rate of the optimizer to ensure convergence and stability of the training process.

Using our novel modified $BCE_Loss + KLDivergenceLoss$ for the Generator leads to faster and smoother training convergence. The Generator loss consists of the binary cross-entropy loss for measuring effectiveness of minimizing Discriminator scores with fake data and the KL Divergence Loss minimization helps in effectiveness of generating fake data closer to original data distribution.

Adding KL divergence loss to the generator may increase the computational cost and training time of the model, as it requires calculating the distributions of the real and fake samples and computing their divergence. Therefore, it was necessary to tune the weighting of the different loss terms and the learning rate of the optimizer to ensure convergence and stability of the training process.

Effect of modified BCE_Loss :

- binary cross entropy loss for fake data
- Generator produces fake data to minimize Discriminator's score on fake data
- minimizing this loss effectively guides Generator in direction of being able to deceive Discriminator by reducing its score

Effect of $KLDivergenceLoss$:

- The Generator learns better to produce samples closely matching the distribution of the real data.
- Helps prevent mode collapse
- Provides a more stable training process
- Allows for better control of the generated data distribution

So for our K-CGAN model setting new loss functions become:

$$CUSTOM\ GAN\ Discriminator\ Loss = Traditional\ D_Loss \quad (35)$$

$$CUSTOM\ GAN\ Generator\ Loss = Traditional\ G_Loss + KLDivergenceLoss \quad (36)$$

a. K-CGAN Discriminator Loss

The Discriminator loss of K-CGAN has two parts: the binary cross-entropy loss for the real data and the binary cross-entropy from the generated data. The first part deals with the accuracy of the Discriminator to identify real data. The Discriminator takes in the original data and creates a prediction, which is compared to the true label. The objective is to lessen the loss term. Moreover, the second term, binary cross-entropy for generated data, measures the efficiency of the Discriminator network to discriminate between real and generated data. The objective is to lessen the loss term.

Furthermore, in the discriminator loss, the gradients and update weights are calculated. This process is known as back-propagation, in which the gradients of the loss with respect to the weights are calculated. The optimizer is utilized to update the weights, ultimately leading to the minimum possible loss.

Mathematical representation of Discriminator (D) loss is minimization of *BCE_Loss*:

$$D_Loss = -\frac{1}{N} \sum_{i=1}^N [\log(D(x_i)) + \log(1 - D(G(z_i)))]$$

which can be equivalently represented as:

$$D_Loss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (37)$$

Where, x_i are training data samples, y_i stands for the actual class labels of training data (total data samples = N [$i \in N$]), the probability of the actual class y_i predicted by the model is represented as $p(y_i)$ or \hat{y}_i . The probability of class one is $p(y_i)$ or \hat{y}_i and the probability of class zero is $1 - p(y_i)$ or equivalently $1 - \hat{y}_i$. The binary cross entropy loss function estimates the average cross entropy of all real and fake data samples, where " y_i " denotes the class label, and " $\log(p(y_i))$ " [or equivalently " $\log(\hat{y}_i)$ "] denotes the predicted log-probability of the data for the i^{th} sample. The **Discriminator loss** consists of the binary cross-entropy loss for the measurement of genuine data and the binary cross-entropy loss for the assessment of false data.

The first term, " $-\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i)]$ " is binary cross entropy loss for the real data, measures how

effectively the Discriminator can accurately identify genuine data. With the help of binary cross-entropy loss, the Discriminator network compares its prediction against the true label after receiving the actual data. Minimising this loss term is the objective.

The second term, " $-\frac{1}{N} \sum_{i=1}^N [(1 - y_i) \log(1 - \hat{y}_i)]$ " is binary cross – entropy loss for the fake data,

measuring the ability of the Discriminator to discriminate between real and false data.

The Discriminator network receives false data generated by the Generator network, which is then fed into it. The Discriminator network's prediction is then compared against the fake label using binary cross-entropy loss. This loss term's duration must also be kept to a minimum for the optimized performance. The objective of training is to reduce the total Discriminator loss, and this can be achieved by summing together its two components. The Discriminator network improves its ability to discern between authentic and fake data by minimising the Discriminator loss.

b. K-CGAN Generator Loss

While the K-CGAN's Generator (G) loss consists of two components: the **binary cross-entropy loss** and the **KL Divergence** which though creates complexity contributes to its overall effectiveness. During Generator training parameters of Discriminator are frozen and hence Discriminator only receives fake input and hence the first term involving (y_i) in BCE_Loss in Equation (22) above is zeroed out and we are dealing with a modified case of BCE_Loss of Generator.

Mathematical representation of our novel Generator (G) loss is minimization of modified BCE_Loss + KLDivergenceLoss:

$$G_Loss = \frac{1}{N} \sum_{i=1}^N [\log(1 - D(G(z_i)))] + D_{KL}(Sample\ Data, Original\ Data) \quad (38)$$

Which can be equivalently represented as:

$$G_Loss = \frac{1}{N} \sum_{i=1}^N \log(1 - \hat{y}_i) + \frac{1}{N} \sum_{i=1}^N [p_{sample}(x_i) \log(\frac{p_{sample}(x_i)}{q_{original}(x_i)})] \quad (39)$$

Where, x_i are training data samples, y_i stand for the actual class labels of training data (total data samples = $N[i \in N]$), the probability of the actual class y_i predicted by the model is represented as $p(y_i)$ or \hat{y}_i . The probability of class one is $p(y_i)$ or \hat{y}_i and the probability of class zero is $1 - p(y_i)$ or equivalently $1 - \hat{y}_i$. The modified binary cross entropy loss function for Generator estimates the average cross entropy over fake data samples, where $\log 1 - \hat{y}_i$ denotes the predicted log-probability of the data for the i^{th} sample. The **Generator loss** consists of the binary cross-entropy loss for the measurement of fake data and the **KL Divergence Loss**.

The first term, " $\frac{1}{N} \sum_{i=1}^N \log(1 - \hat{y}_i)$ " or equivalently " $-\frac{1}{N} \sum_{i=1}^N (1 - y_i) \log(1 - \hat{y}_i)$ " for fake data,

[when $y_i = 0$] is modified case of the binary cross entropy loss part for the fake data. Moreover,

the second term, " $\frac{1}{N} \sum_{i=1}^N [p_{sample}(x_i) \log(\frac{p_{sample}(x_i)}{q_{original}(x_i)})]$ " is the **KL Divergence loss** measuring how

effectively Generator is able to generate fake samples close to the distribution of original data samples $p_{sample}(x_i)$ and $q_{original}(x_i)$ respectively represent the probability distributions for sample fake and original data. Minimization of this loss term helps Generator in being able to generate fake data closer to the original data distribution. The degree to which the Generator may deceive the Discriminator is measured by its effectiveness of generating fake data closer to the original data distributions. This way the Generator can deceive the Discriminator in making it feel that fake data are also the real data. The Generator network produces fake data that is more challenging for the Discriminator to identify from actual data in order to maximise this loss term. Minimization of **the second term, KL Divergence loss** helps Generator in that direction by better data sampling. Minimization of **the first term, binary cross-entropy loss** basically helps Generator to maximise its score of fooling Discriminator. The objective is to maximise the binary cross-entropy loss while also minimising the KL divergence, and the Generator loss is a mixture of these two

components. In this way, the Generator network acquires the ability to generate more realistic fake data thereby leading to better performance.

3.2.4 Advantages of using Novelty Loss

When compared to other GANs such as the Wasserstein GAN (WGAN), Non-Saturating GAN (NS GAN), and Least Squares GAN (LS GAN), employing our Custom loss along with the optimised custom hyperparameters have the following advantages:

Better control over generated samples: Novelty loss enables us to add additional constraints on generated samples, such as KL-divergence, which allows for better control over the quality and diversity of generated samples.

Enhanced Robustness: By incorporating KL-divergence and binary cross-entropy components, the custom novelty loss enhances the robustness of the K-CGAN against noisy or imperfect data. It helps the generator focus on capturing the salient features of the data distribution, making it less sensitive to noisy or outlier samples during training.

Improved Convergence: The custom novelty loss encourages the generator to produce samples that are both diverse and representative of the underlying data distribution. This balanced training objective aids in achieving faster and more stable convergence during training, reducing the likelihood of mode collapse or oscillations commonly observed in other GANs.

Reduced Overfitting: The inclusion of dropout layers in the custom novelty loss architecture aids in reducing overfitting, allowing the generator to learn more generalizable features from the data and avoid memorising specific data instances.

More stable training: The use of KL-divergence and binary cross-entropy in novelty loss leads to more stable training compared to other GANs.

Improved Generalisation: The custom novelty loss encourages the generator to explore a larger region of the data distribution, resulting in improved generalization to unseen data samples. This can be particularly beneficial when working with limited training data or in scenarios where data variability is high.

More effective in learning complex distributions: Utilising a custom loss, backed by the optimal K-CGAN hyperparameters, proves to be highly effective in capturing intricate distributions. This approach empowers the generation of a vast array of diverse samples, enhancing the learning process significantly.

3.2.5 Disadvantages of using Novelty Loss

In comparison to other GAN loss methods, adopting Custom loss may have the following drawback and practical challenges that one may encounter:

High computational cost: Compared to other GAN loss functions, the Custom loss function is more computationally expensive and time-consuming to implement. This is due to the computation of binary cross-entropy and KL divergence, which can take a while for big datasets or complicated models.

Sensitivity to hyperparameters: The custom loss function, like other GAN loss functions, could be sensitive to the selection of hyperparameters, such as the learning rate, batch size, and number of epochs and all other hyperparameters listed earlier (Table 1 and 2). Finding the best settings for these hyperparameters might be difficult as a result, and significant hyperparameter tuning may be necessary.

Optimization Difficulties: The presence of multiple loss components in the custom loss can create challenges in optimizing the generator and discriminator simultaneously. Fine-tuning the weightings of the loss components to strike the right balance becomes an additional optimization task.

It's essential to keep in mind the practical challenges that come with implementing GANs. High computational cost, sensitivity to hyperparameters, and optimization difficulties are some of the potential obstacles that may arise. Therefore, it's crucial to carefully consider these challenges and find ways to mitigate them when using custom loss in GAN experiments.

3.3 The Discriminator and the Generator Architectures of K-CGAN

The proposed algorithm of our method is further described in the Figure 6 K-CGAN discriminator and generator architectures. The network architecture and layer transformation with input-output dimensionality is explained in Figure 7 (a), (b) and (c).

Data Info:

- x (original input sample space) $\in R^m$ [$m = 29$]
 - for our case, we are using data with 29 features
- z (Noise Input - Latent Space) $\in R^d$ [$d = 100$]
 - for Noise Input we use random vectors of dimension 100
- y (class labels) $\in R^1$ [# class labels = 2; 0 or 1]
 - label 1 for real samples, label 0 for fake samples
- b
 - batch size during model training
- Training Samples
 - to overcome the imbalanced nature of dataset due to less frequency of fraudulent transactions in the training dataset, we used SMOTE method for oversampling fraudulent transactions to make training dataset balanced
 - we used this method only for sampling during training and we do not use it during inferencing

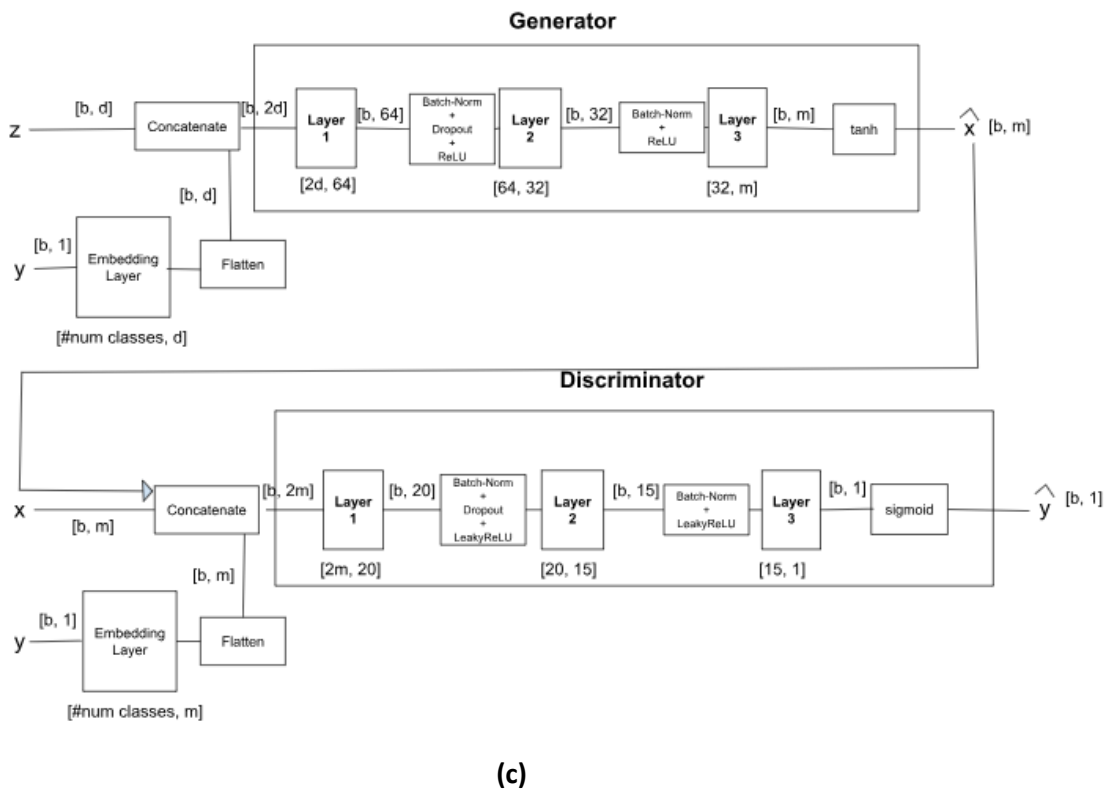
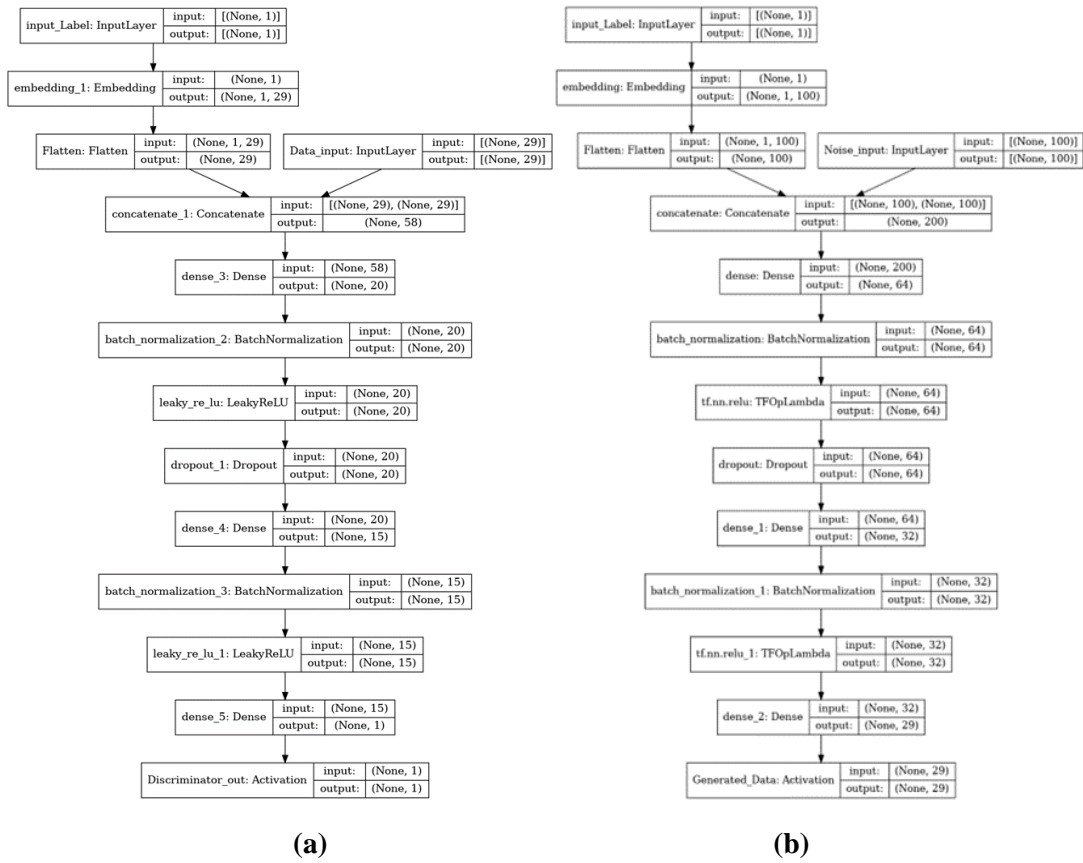


Figure 7: TensorFlow’s representation of K-CGAN: (a) Discriminator; (b) Generator and (c) Network Layers & Training Architecture

Each layer in a neural network effectively does a matrix multiplication and addition on matrix inputs and converts it to suitable matrix outputs e.g.:

- layer 1 of Generator takes a matrix of shape $[b, 2d]$ and then:
 - multiplies it with matrix of weights of dimension $[2d, 64]$ to produce an output of $[b, 64]$
 - then a matrix of biases of shape $[b, 64]$ gets added to above output to produce another output of $[b, 64]$
 - then a batch normalisation operation normalises values of this $[b, 64]$ values for smoother training
 - an additional dropout operation may be done to further reduce overfitting
 - and finally an activation function like ReLU or tanh or sigmoid or Leaky ReLU gets applied to these $[b, 64]$ values before getting passed onto next layer

Similar matrix transformations and operations happen in layer 2, layer 3 ... of any neural network.

An equivalent representation of Figure 7 (c) by TensorFlow library's model architecture print is given in Figure 7 (a) and (b). TensorFlow library uses None in place of b for Batch_Size as Batch_Size is not an intrinsic network variable but a training choice, so the architecture remains valid for any Batch_Size (b). Network Architecture of Discriminator and Generator from TensorFlow are given in Figure 7 (a) and Figure 7 (b) respectively.

The K-CGAN Discriminator architecture in Figure 7 (a) encompasses a series of pivotal layers that play a vital role in data processing and classification and K-CGAN model performance. Within the Discriminator's architecture, an input layer accepts label data with a single neuron. Subsequently, an embedding layer encodes this input data, resulting in a dimension of (1, 100). The flatten layer reshapes this to (100). The primary noise input layer, with 100 dimensions, accepts stochastic noise data. These inputs are concatenated in a layer, resulting in a combined representation with dimensions (200). A dense layer with 64 neurons is employed, followed by batch normalization and a rectified linear unit (ReLU) activation function. A dropout layer with 64 neurons is applied to mitigate overfitting. Another dense layer with 32 neurons follows, along with batch normalization and a ReLU activation. A final dense layer with 29 neurons is used, and an activation layer determines the Discriminator's output, which is represented as (29) dimensions. This latter layer performs binary classification, distinguishing between generated and real data. The K-CGAN Discriminator architecture, marked by its intricate layers, ensures accurate classification of generated data in alignment with predefined criteria.

Further, the K-CGAN's Generator architecture in Figure 7 (b), multiple layers collaboratively produce the desired output and performance of the model. In the Generator's architecture, an input layer with a single neuron serves as the entry point for label information. This is followed by an embedding layer that transforms the input data, resulting in an output dimension of (1, 29). Subsequently, a flatten layer reshapes this data to (29). The primary data input layer accepts data with a dimension of (29). Following this, a concatenation layer merges the processed label data and primary data input, resulting in a combined representation of (58) dimensions. A dense layer with 20 neurons is applied, followed by batch normalization and a leaky ReLU activation function. A dropout layer with 20 neurons follows to prevent overfitting. Another dense layer with 15 neurons is employed, followed by batch normalization and a leaky ReLU activation. Finally, a final dense layer with a single neuron produces the generator's output, and an activation layer determines the synthetic data. Each layer in this architecture contributes significantly to generating realistic data closely resembling the original input.

As indicated in Algorithm 2 the initial stage of training the K-CGAN involves generating Random Noise with batch size and latent dimension, where the batch size is the size of the input batch of data and the latent dimension is the dimension of the noise. Afterwards the Generator generates fake data. The Generator of novelty K-CGAN transforms the random noise and labels into fake data. On the other hand, the Discriminator

links both the real data and the generated synthetic data with their labels to shape the combined labels and combined data. Our proposed framework then prepares target labels to form a binary label that discriminates real data from fake data.

The next phase of K-CGAN involves training the Discriminator. In this phase, the Discriminator is used to classify the combined labels and data to minimise the Discriminator loss, defined by the loss function between the predictions and true labels. Moreover, the gradients of the loss with respect to the Discriminator's trainable weights were computed and used to update the parameters of the Discriminator. The Discriminator's loss was calculated in the next stage of training. The Discriminator loss is the binary cross-entropy between misleading labels and prediction. Furthermore, the gradients and update weights were calculated. This process is known as back-propagation, in which the gradients of the loss with respect to the weights are calculated. The optimizer was then utilised to update the weights, ultimately leading to the minimum possible loss.

The next steps of training novelty K-CGAN involve the preparation of data and the start of the Generator training. In this training phase, the Generator is trained to transform the noise and real labels into fake data, and uses the Discriminator to classify these fake data samples. Moreover, the Generator aims to maximise the loss defined by the loss function between the misleading labels and the Discriminator's predictions of the fake data. The Generator loss was calculated after the Discriminator loss. The Generator loss consists of two terms: binary cross-entropy between true labels and predicted labels by the Discriminator, and KL Divergence between the original data and fake data generated by the Generator. The Kullback-Leibler (KL) divergence measure of the difference between two probability distributions. This is used to optimise the parameters of the Discriminator model based on the difference between the original data distribution and the fake data distribution of the Generator model. The next stage of training the K-CGAN is referred to as backpropagation (Li and Kang, 2022) where the gradients of the loss with respect to the weights are calculated. The optimizer was then utilised to update the weights, ultimately leading to the minimum possible loss. In the last step of training, tracking of the Discriminator and the Generator loss is performed using the Generator loss tracker and Discriminator loss tracker objects, returning these losses as a dictionary. The feature selection procedures are summarised in Algorithm 1. Further, the training details of K-CGAN are summarised in Algorithm 2.

Algorithm 1 Feature Selection Procedures

```
1: Epochs: 10000
2: Batch_size = 64
3:     #How Many Data in one Batch
4: HP_NOISE = 100
5:     #Lengh of Noise Vector
6: HP_DROPOUT = 0.2
7:     #Dropout to be used in the Neural Network
8: HP_WEIGHTS_INIT = 'glorot_uniform'
9:     #weight Initialization Training phase
10: HP_DISCRIMINATOR_LAYERS = '20, 15, 1'
11:     # defined three possibilities of hidden layers
12: HP_GENERATOR_LAYERS = '64, 32, 29'
13:     #Here we have defined three possibilities of hidden
14:     layers for the Generator model.
15: SAMPLES_COUNT = 400
16:     #How Many fake samples to be generated while
17:     calculating F1 Score
17: EARLY_STOPPER_PATIENCE = 50
18:     #Stop Training if accuracy doesn't improved for
19:     continuous n epochs
```

Algorithm 2 Training procedure using K-CGAN

```
1: Generate Random noise (noiseD)
2:     with shape (batch_size, 2:self.latent_dim),
3:     (batch_size)=size of the input batch of data
4:     (latent_dim) =dimension of the noise.
5: Generate fake data
6:     #using the Generator to transform the random
7:     noise (noise) and (real_labels) into fake data.
8: Data preparation for the Discriminator
9: Training phase
10:    #The real data (real_data) and generated fake
11:    data
12:    (generated_data) along with their labels
13:    (real_labels) to form (combined_data) and
14:    (combined_labels)
15:    Target labels are prepared
16:    Binary labels (labels) are formed
17: Use discriminator to classify combined_data and
18: combined_labels
19:    The goal is minimize the loss (d_loss)
20:    Prepare Data for Generator Training
21: Generator training during training
22:    Random noise (noiseG) is generated with
23:    (batch_size, self.latent_dim) to train the generator
24: Uses generator
25:    #transform noise (noiseG) and labels
26:    (real_labels) into (fake_data)
27: Use discriminator to classify these fake data.
28:    #the generator aims to maximize the loss (g_loss)
29:    #the Generator Loss is equal to BCE + KL
30: Calculations
31: #Backpropagation process: gradients of the loss
32: with respect to the weights are calculated
33: Goal is minimum loss
34: Tracking step
35:    (g_loss) and (gen_loss) tracking using
36:    (gen_loss_tracker) and (disc_loss_tracker)
37:    #Returns these losses as dictionary
```

Chapter 4

Novelty Loss development: Implementation and Experiments using Multiple Methods

4.1. Introduction

This section provides a comprehensive overview of the implementation and experiments conducted on two datasets, Credit Card Fraud (Kaggle, 2021) and Breast Cancer Wisconsin (Diagnostic) (WBCD) (Wolberg et al., 1992). Experiments conducted utilising several GAN-based architectures such as Conditional GAN (Mirza & Osindero, 2014), Wasserstein GAN (Arjovsky et al., 2017), Least Squares GAN (Mao et al., 2017), Non-Saturating GAN (Shannon et al., 2020), and Semantic Divergence GAN (Charitou et al., 2021). To evaluate the quality of the dataset, we employed various methods, including classification performance, cosine similarity approaches, bivariate and univariate correlations. Furthermore, in order to maximise performance, the research utilised hyper-parameter tuning to identify the optimal hyper-parameters of these architectures. In order to ensure model accuracy and performance, an in-depth quality assurance validation has been conducted to validate K-CGAN performance on the credit card fraud and breast cancer datasets. To evaluate the efficacy of K-CGAN generated synthetic data, the existing oversampling strategies such as ADASYN (He et al., 2008), SMOTE (Chawla et al., 2002), and B-SMOTE (Han et al., 2005) as well as GAN based techniques such as Conditional GAN (Mirza & Osindero, 2014), Wasserstein GAN (Arjovsky et al., 2017), Least Squares GAN (Mao et al., 2017), Non-Saturating GAN (Shannon et al., 2020), and Semantic Divergence GAN (Charitou et al., 2021) have been implemented to compare their performance.

Finally our experiments also demonstrate the introduction of a novel loss function that incorporates KL divergence loss combined with Binary Cross entropy to ensure both distributions are close to each other. Hyper-parameter tuning to identify the optimal weight for the KL divergence loss also presented. By utilising this methodology, various GAN architectures and oversampling techniques have been compared in order to find the most effective approach for creating synthetic data related to credit card fraud transactions and breast cancer data. Additionally, incorporating a novelty loss function improved the performance of the GAN-generated synthetic data and demonstrated enhanced performance of popular classification methods such as XGBoost (Chen & Guestrin, 2016), Random forest (Breiman, 2001), K-Nearest Neighbor (Cover & Hart, 1967), Multilayer Perceptron (Rosenblatt, F., 1957), and Logistic regression (DeMaris, A., 1995).

4.2 Datasets Pre-processing and Architectures

Credit Card Fraud dataset

To evaluate the performance of our method, we utilised credit card fraud dataset as our first dataset for the experiments. We accessed a publicly-available dataset from Kaggle, containing credit card transaction data of European consumers collected in September 2013 (Kaggle, 2021). The credit card fraud dataset is widely used in the field of machine learning for detecting fraud (Alharbi et al. 2022). The imbalanced dataset consists

of 284,807 transactions with a number (492) being fraudulent, resulting in an unequal distribution; each has 30 features - such as 'Time' and 'Amount' – accompanied by 28 anonymized attributes (V1 to V28). This database is free of outliers and missing values.

In order to enhance a GAN's training process the features were scaled between -1 and +1 using Min-Max scaling (Smith, 2020), as it reduces computational cost while also making the optimizer more efficient at locating the solution enabling GAN to converge faster. Subsequently, all columns were then converted to float32 type, which brings numerous advantages such as improved speed of execution and accuracy in representing real numbers.

Features columns: The 'Time' column was removed because the chances of a credit card fraud happening at any specified time or day are very slim. Therefore, there is no need to monitor this factor with high frequency. Consequently, the 'Time' column is not very helpful when it comes to uncovering fraudulent patterns. As a result, the following columns were used as features V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, V28 and 'Amount'.

Target Column: The objective of this data is to distinguish between fraud and genuine transactions through the target 'Class' column, which can hold two values: 1 for fraudulent occurrences and 0 when a transaction is valid. Our GAN model's generator is trained to produce fake data conditioned on the target class labels.

Wisconsin Diagnostic Breast Cancer (WDBC) dataset

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset is a well-known and accessible set of information about breast cancer tumours, available on the UCI machine learning Repository (Wolberg et al., 1992; Wolberg et al., 1995). This dataset is widely used in the machine learning field for detecting breast cancer (Kabir and Ludwig, 2018; Sharma et al., 2018). William H. Wolberg gathered this data at the University of Wisconsin Hospital, Madison back in the early 1990s. There are 569 instances in the dataset representing tumour samples. Out of these, 357 are classified as benign and 212 as malignant.

Features columns: The dataset includes 30 numerical features that measure the characteristics of the cell nuclei in each sample. These features include mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, mean symmetry, mean fractal dimension, as well as their standard errors and worst values.

Target Column: The 'Diagnosis' is the target variable and it can either be benign (not cancerous) or malignant (cancerous). These values are represented by 0 and 1 respectively.

4.3 Experimental Settings

The following libraries were used for implementation:

- a) numpy: A library for numerical computing in Python, used for handling arrays and mathematical operations.
- b) pandas: A data manipulation library, used for reading and processing tabular data.
- c) tensorflow: An open-source machine learning framework developed by Google, used for building and training deep learning models.
- d) matplotlib: A plotting library, used for creating visualizations of the data and model performance.
- e) seaborn: A data visualization library based on matplotlib, used for creating more plots.
- f) sklearn: The scikit-learn library, for data preprocessing, model evaluation.

- g) keras: A high-level neural networks API built on top of tensorflow, used for defining and training neural network models.
- h) MinMaxScaler: A class from sklearn.preprocessing, used for scaling the data to a specific range (e.g., [0, 1]).
- i) train_test_split: A function from sklearn.model_selection, used for splitting the data into training and testing sets.
- j) Adam: An optimization algorithm from tensorflow.keras.optimizers, used for training the generator and discriminator networks.
- k) Dense: A class from tensorflow.keras.layers, used for creating fully connected (dense) layers in the neural networks.
- l) LeakyReLU: A class from tensorflow.keras.layers, used as the activation function for the discriminator network.
- m) ReLU: A class from tensorflow.keras.layers, used as the activation function for the generator network.
- n) Dropout: A class from tensorflow.keras.layers, used for applying dropout regularization to the neural networks.
- o) BinaryCrossentropy: A loss function class from tensorflow.keras.losses, used for calculating the binary cross-entropy loss during training.
- p) KLDivergence: A loss function class from tensorflow.keras.losses, used for calculating the Kullback-Leibler (KL) divergence loss during training.
- q) GlorotUniform: An initializer class from tensorflow.keras.initializers, used for initializing the weights in the generator network.
- r) L2: A regularizer class from tensorflow.keras.regularizers, used for applying L2 regularization to the neural networks.

In order to assess the efficiency of various oversampling techniques on a skewed dataset of credit card fraud transactions and breast cancer detection, multiple experiments have been conducted. Several GAN-based architectures were implemented as oversampling methods, including: NS GAN (Shannon et al., 2020), Vanilla cGAN (Mirza & Osindero, 2014), SDG GAN (Charitou et al., 2021), LS GAN (Mao et al., 2017), and WGAN (Arjovsky et al., 2017), Novelty Loss K-CGAN and Novelty Loss RNN.

Further oversampling techniques were implemented including: ADASYN (He et al., 2008), SMOTE (Chawla et al., 2002), and B-SMOTE (Han et al., 2005). In order to evaluate the precision, recall and F1 scores of the methods, several classification methods were utilised, including: XGBoost (Chen & Guestrin, 2016), Random forest (Breiman, 2001), K-Nearest Neighbor (Cover and Hart, 1967), Multilayer Perceptron (Rosenblatt, F., 1957), and Logistic regression (DeMaris, A., 1995).

4.3.1 Hyperparameter Settings of GAN-based Oversampling Methods

Table 3 shows the initial hyperparameters settings of three popular GAN methods: Vanilla CGAN, WGAN, and NS GAN. It is imperative to mention that GANs are highly sensitive to the hyperparameter settings used during training. Activation function is an important hyperparameter that defines the output range of the neurons and plays a critical role in determining the convergence of the network. The Vanilla cGAN uses Leaky ReLU activation (Maas et al., 2013) function in both the generator and discriminator networks. Overall Leaky ReLU is known to provide better gradients compared to other activation functions, which is essential for efficient training. WGAN employs a different activation function, namely the Rectified Linear Unit (ReLU) function (Nair and Hinton, 2010), which has become the de-facto choice for activation functions in deep learning. ReLU is computationally efficient and has shown good performance in GANs, although there might be issues with dead neurons in certain scenarios. NS GAN, on the other hand, uses the

LeakyReLU activation function in the discriminator and generator networks. LeakyReLU has the advantage of providing better gradients and is known to help avoid the “dying ReLU” problem. Another important hyperparameter is the batch size, which determines the number of samples used in each training iteration. In Vanilla cGAN, the batch size is set to 32, while NS GAN and WGAN use a batch size of 64 and 16, respectively. A larger batch size generally leads to better convergence with a more accurate gradient estimation, but also requires more memory and computation. Dropout (Srivastava et al., 2014) is a regularisation technique that helps prevent overfitting during training. In Vanilla CGAN, a Dropout probability of 0.5 is used, while in WGAN and NS GAN, the Dropout probabilities are 0.1 and 0.5, respectively. Another important hyperparameter is the optimizer, which determines how the model updates weights in response to the loss gradient. In Vanilla CGAN, RMSProp optimizer (Tieleman and Hinton, 2012) is used, while WGAN uses the AdaGrad optimizer (Duchi et al., 2011), and NS GAN uses the Adam optimizer (Kingma and Ba, 2014). These optimizers have different strengths and are chosen based on the specific requirements of the GAN model. The learning rate is used to control the step size in gradient descent while updating the model (Goodfellow et al., 2016). In all three models, the learning rate is set to 0.001, 0.001, and 0.0001 for WGAN, NS GAN, and Vanilla CGAN, respectively. Finally, the number of layers in the generator and discriminator networks are also important hyperparameters. In all three models, the architecture is similar, with three layers in the discriminator network (128, 64, and 32 neurons) and two layers in the generator network (64 and 32 neurons).

Table 3: Vanilla CGAN, WGAN and NS GAN hyperparameter settings

Hyperparameter	Vanilla CGAN	WGAN	NS GAN
Activation	Leaky ReLU	Relu	LeakyReLU
Batch Size	32	16	64
Dropout	0.5	0.1	0.5
Optimizer	RMSProp	AdaGRAD	Adam
Learning Rate	0.0001	0.001	0.001
Discriminator Layers	128,64,32	128,64,32	128,64,32
Generator Layers	64,32	64,32	64,32

Further, Table 4 presents the initial hyperparameter settings for LS GAN, SDG GAN and Custom K-CGAN. The activation function used for the models was LeakyReLU. However, the batch size for LS GAN and Custom K-CGAN was set to 32, while for SDG GAN, it was set to 64. We applied dropout regularisation to models to prevent overfitting. Here, we used a dropout rate of 0.5 for LS GAN and Custom K-CGAN and 0.1 for SDG GAN. To optimise the models, we employed the RMSProp optimizer. In terms of learning rate, we used 0.0001 for LS GAN and Custom K-CGAN and 0.001 for SDG GAN. LS GAN and Custom K-CGAN had three discriminator layers (128, 64, and 32), while SDG GAN had more layers (256, 128, and 64). On the other hand, all models had similar generator layers - 64 and 32.

Table 4: LS GAN, SDG GAN and Custom K-CGAN initial experimental hyperparameter settings

Hyperparameter	LS GAN	SDG GAN	Custom K-CGAN
Activation	LeakyReLU	LeakyReLU	LeakyReLU
Batch Size	32	64	32
Dropout	0.5	0.1	0.5
Optimizer	RMSProp	RMSProp	RMSProp
Learning Rate	0.0001	0.001	0.0001
Discriminator Layers	128,64,32	256,128,64	128,64,32
Generator Layers	64,32	64,32	63,32

4.3.2 Hyperparameter Settings of Oversampling Methods

Apart from GAN-based models, the study employed three popular oversampling techniques, namely ADASYN (He et al., 2008), SMOTE (Chawla et al., 2002) and B-SMOTE (Han et al., 2005). These methods are commonly used in the field of machine learning to address the issue of class imbalance in datasets. In all three methods the default settings were used to ensure reproducibility of the results. These oversampling techniques were applied to our dataset to address the class imbalance issue and improve the overall performance of our machine learning models. The settings chosen for each method were based on their default implementation, which are widely accepted in the research community and have been shown to achieve reliable results in practice. SMOTE uses a k-nearest neighbor approach to create synthetic minority points, making use of random oversampling. The default number of nearest neighbors is set to 5 (imbalanced-learn.org, n.d.). ADASYN works similarly to SMOTE by creating synthetic minority examples based on the distribution of majority and minority classes in the data set. The default number of nearest neighbors is 5 and the default ‘synthetic’ points per minority class sample is set to 10 (imbalanced-learn.org, n.d.). B-SMOTE works differently from SMOTE and ADASYN by creating minority points using a borderlinesmote approach. The default number of nearest neighbors for B-SMOTE is 5, the default ‘synthetic’ points per minority class sample is set to 10, and the maximum number of synthetic points that can be generated is 20 (imbalanced-learn.org, n.d.). Table 5 shows the settings of these oversampling methods.

Table 5: SMOTE, ADASYN and B-SMOTE hyperparameter settings

Method	Settings
SMOTE	default number of nearest neighbors is 5 (imbalanced-learn.org, n.d.)
ADASYN	default number of nearest neighbors is 5 and the default ‘synthetic’ points per minority class sample is set to 10 (imbalanced-learn.org, n.d.)
B-SMOTE	default number of nearest neighbors for B-SMOTE is 5, the default ‘synthetic’ points per minority class sample is set to 10, and the maximum number of synthetic points that can be generated is 20 (imbalanced-learn.org, n.d.)

4.3.3. Hyperparameter Settings of Classification Methods

The study also used various classification methods to determine the best approach for upgrading accuracy of classification models when working with an imbalanced dataset of credit card fraud transactions. Implementation consisted of various experiments aimed at determining the optimal oversampling approach and set of hyperparameters. This approach aims at enhancing the accuracy of classification models in the presence of an imbalanced dataset of credit card fraud transactions. It is essential to note that credit card fraud is a major problem globally, and it has adverse impacts on individual victims and the overall economy, hence determining the most effective classification method is essential. The classification methods settings implemented in our study are shown in the following Table 6 for all five classifiers Random Forest, XGBoost, KNearest Neighbor, MLP, and Logistic Regression. The Random Forest and XGBoost methods were set with a random state of 42, while the MLP method was set with a random state of 1 and a maximum iteration cap of 300. The KNearest Neighbor method was set with a n_neighbors value of 100, and the Logistic Regression method was set to default.

Table 6: Classification methods hyperparameter settings

Method	Settings
Random Forest	To control randomness of the sample random_state was set to 42, by setting the default value we're ensuring that the data is getting arranged the same way, as a result it returns the same training and testing subsets. (scikit-learn.org, n.d.)
XGBoost	To control randomness of the sample random_state was set to 42 (scikit-learn.org, n.d.)
KNearest Neighbor	The tuning hyper parameter n_neighbors was set to 100 (scikit-learn.org, n.d.)
MLP	The max_iter parameter represents the maximum number of epochs for model training. It is referred to as "maximum" because the learning process may stop before reaching the maximum number of iterations, depending on other termination criteria, we have set it to 300. To control the random factor (random_state) was set to 1. It's recommended to set the seed for the random generator to confirm that the outcomes can be consistently reproduced. random_state=1, max_iter=300 (scikit-learn.org, n.d.)
Logistic Regression	The default settings were follows: solver: 'lbfgs', max_iter: 100, multi_class: 'auto', n_jobs: None, and random_state: None. (scikit-learn.org, n.d.)

4.3.4 Original and Balanced Datasets using Oversampling Techniques

a. Original imbalanced dataset

The original credit card dataset is depicted in Table 7 exhibits a significant imbalance in the distribution of fraudulent and valid transactions.

Table 7: Original imbalanced credit card fraud dataset

Description	Value
Valid Transactions	284315
Fraudulent Transactions	492

Specifically, out of a total of 284,807 transactions, 284,315 transactions are valid while only 492 transactions are fraudulent. This disparity highlights the complexity of detecting fraudulent transactions and the importance of developing robust algorithms that can accurately identify such cases. The presence of such a highly imbalanced dataset poses a significant challenge to the construction of predictive models that can accurately classify credit card transactions as valid or fraudulent. One of the significant issues that arise in imbalanced datasets is that standard classification algorithms tend to focus more on the majority class, which, in this case, is the valid transaction category. This is a major issue in detecting fraudulent transactions, which form a small minority of the total transaction volume. Therefore, the development of efficient algorithms that can adequately identify fraudulent transactions and avoid bias towards the majority class is crucial for efficient fraud detection in the financial industry.

b. Balanced credit card dataset using data augmentation

Moreover, Table 8 depicts a balanced credit card fraud dataset upon oversampling using SMOTE, B-SMOTE, ADASYN as well as GAN-based methods. These three oversampling methods utilised helped in creating balanced credit card fraud dataset by artificially increasing the number of fraudulent transactions, thereby improving the accuracy and efficiency of models developed for detecting fraudulent credit card transactions. The dataset comprises 284,315 valid transactions and an equal number of fraudulent transactions, which creates a balanced class distribution. Credit card fraud is a significant issue, and detecting fraudulent transactions accurately is critical for maintaining financial integrity and protecting consumers from monetary loss. These methods have been used to artificially increase the number of fraudulent transactions in the dataset to create a more balanced class distribution.

Table 8: Balanced credit card fraud dataset by oversampling the minority class with SMOTE, B-SMOTE and ADASYN and GAN based methods

Description	Value
Valid Transactions	284315
Fraudulent Transactions	284315

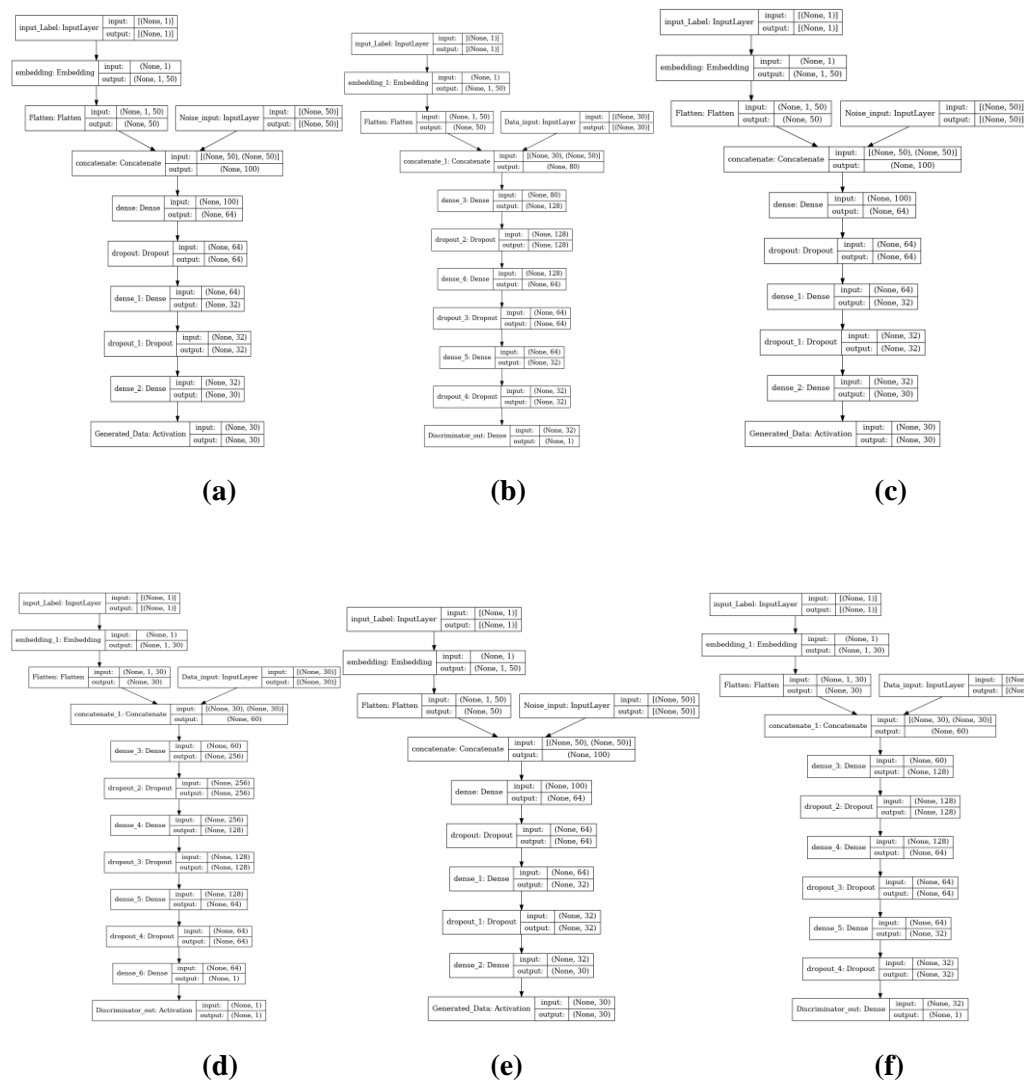
The SMOTE oversampling method was used to generate new fraudulent transactions by creating synthetic samples that are similar to the existing ones but with small variations. B-SMOTE oversampling technique improves upon SMOTE by using random samples from the data to initialise a generation of interactions to avoid overfitting the training model. Similarly, the ADASYN oversampling technique is designed with the

aim of reducing overfitting by creating additional minority class samples that are more difficult to learn by increasing the density of the learned distribution around the borderline. It is noteworthy to mention that oversampling with GAN based data augmentation methods we implemented to overcome the class imbalance problem in machine learning models. In this approach, a generator network is trained to create synthetic data that resembles the minority class distribution. Then, a discriminator network evaluates the quality of the generated samples and provides feedback to the generator. As these networks compete against each other, the generator becomes better at creating realistic samples, thus increasing the diversity and quantity of the minority class data, thus producing a more balanced class distribution.

4.3.5 Generator and Discriminator Architectures of GAN based Methods

This section offers a detailed explanation about the training progress of the methods to oversample the minority class as shown in previous Tables 7 and 8 for the methods. In our experiments we've compared the performance of various GAN-based methods on the credit card dataset.

Architectures



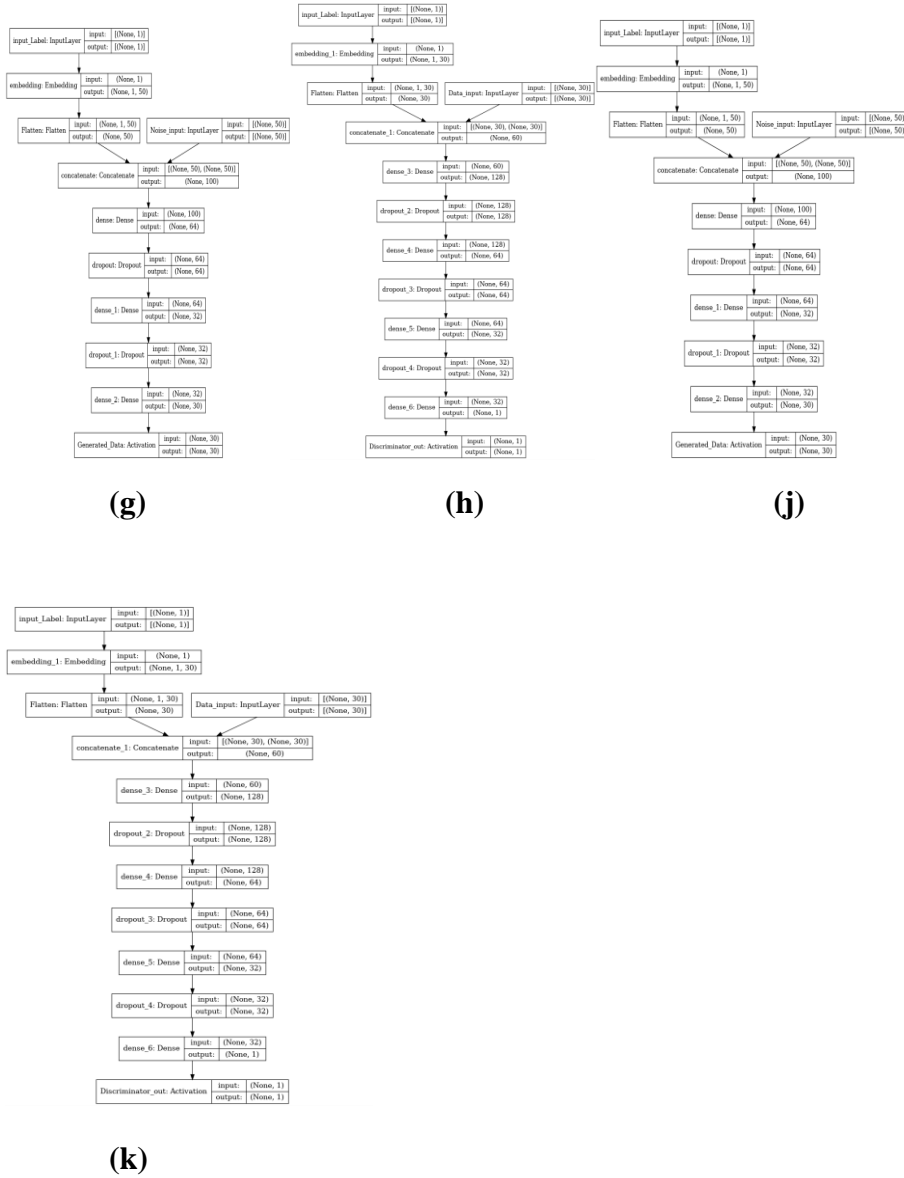


Figure 8: The WGAN Generator (a) and Discriminator (b), SDG GAN Generator (a) and Discriminator (b), NS GAN Generator (e) and Discriminator (f), LS GAN Generator (g) and Discriminator (h), K-CGAN Generator (j) and Discriminator (k)

The term "None" in the input shape represents that the network can accept a variable number of samples in each batch during training. It is advantageous to allow for variable batch sizes in GANs where the generator and discriminator need to handle both real and generated samples during training. When training a neural network, the batch size is the number of samples that will be processed together in one forward and backward pass.

In the following experimentation we've implemented the WGAN method (Arjovsky et al., 2017) to oversample the minority class transactions. The WGAN Generator architecture (depicted in Figure 8 (a)) encompasses an input layer, an embedding layer, a flatten layer, a noise input layer, a concatenate layer, two dense layers with dropout layers (Srivastava et al., 2014) in between, and a final activation layer. The input layer takes a vector of size (1) and produces a vector of the same size. The embedding layer processes this vector, resulting in a vector of size (1, 50) which is then flattened to (50). A noise input layer maintains the same dimensions as the flattened embedding vector. The concatenate layer combines the flattened embedding

vector and the noise input vector, yielding a vector of size (100). This vector is fed into a dense layer, producing a vector of size (64), followed by a dropout layer. Another dense layer generates a vector of size (32), succeeded by another dropout layer. The final dense layer outputs a vector of size (30), which is passed through an activation layer to generate the final synthesised data. The WGAN Discriminator architecture (shown in Figure 8 (b)) comprises an Input_Label layer taking a single input of size (1) and producing the same size output. An Embedding Layer processes this output, converting it to size (1, 50), and the Flatten layer further reduces it to size (50). In parallel, a Noise_Input layer accepts a single input of size (30) and outputs the same size output. The Concatenate layer combines the outputs of the Noise_Input layer and the Flatten layer, resulting in an output of size (80). A Dense layer processes this output, yielding size (128), followed by a Dropout layer. Another dense layer with 64 neurons follows, succeeded by a Dropout layer. Finally, a Dense layer with 32 neurons is used, followed by another Dropout layer. The Discriminator_out layer accepts the output of the last Dropout layer and produces a single value of size (1).

In the following experimentation we've implemented the SDG GAN method (Charitou et al., 2021). The architecture of the SDG GAN Generator (illustrated in Figure 8 (c)) involves sequential layers that collaborate to generate synthetic data. The process starts with an Input Layer, which accepts a shape of (1) and outputs the same shape. Subsequently, an Embedding Layer processes this input, resulting in an output of shape (1, 50). The output is then flattened using a Flatten Layer, transforming it into a shape of (50). Next, a Noise_Input Layer processes the input of shape (50), maintaining its dimensions. A Concatenate Layer combines the outputs of the Flatten Layer and the Noise_Input Layer, resulting in an output shape of (100). This is directed to a Dense Layer, generating an output shape of (64), which is then subjected to a Dropout Layer. A subsequent Dense Layer processes the output of the previous Dropout Layer, resulting in an output shape of (32), followed by another Dropout Layer. Finally, a Dense Layer generates an output shape of (30), which is then passed through an Activation Layer to yield the final synthesised data of shape (30). The architecture of the SDG GAN Discriminator (depicted in Figure 8 (d)) comprises a series of interconnected layers aimed at assessing the authenticity of the generated data. Starting with an input layer, the input shape is transformed from (1) to the same shape of (1) in the output. An Embedding Layer then processes this input, generating an output shape of (1, 50), followed by a Flatten Layer that further flattens it to a shape of (50). In parallel, a Noise_Input Layer processes input of shape (30), producing an output of the same shape. A Concatenate Layer merges the outputs of the Noise_Input Layer and the Flatten Layer, resulting in an output shape of (80). Two Dense Layers follow, with input shapes of (80) and (128), generating output shapes of (128). These dense layers are interspersed with four Dropout Layers, each of which maintains the same dimensions as the input layers. Finally, the Discriminator_out layer processes the output of the last Dropout Layer, producing an output shape of (1).

The NS GAN (Shannon et al., 2020) Generator architecture depicted in Figure 8 (e) consists of a sequence of interconnected layers responsible for generating data based on the provided input. This architecture initiates with an Input Layer, which accepts the label and produces an output shape of (1). This is immediately followed by an Embedding Layer that takes the output from the Input Layer and generates a shape of (1, 50). The resulting output is then flattened into shape (50). Subsequently, a Noise_Input Layer processes the input of shape (50) and maintains the same shape. The outputs from both the Input Layer and the Noise_Input Layer are united through a Concatenate Layer, yielding an output shape of (100). This output then undergoes processing by a Dense Layer, resulting in an output shape of (64), which is then subjected to a Dropout Layer. Another Dense Layer follows, processing the output of the previous Dropout Layer to yield an output shape of (32), which in turn encounters another Dropout Layer. Finally, a last Dense layer processes the output, resulting in a shape of (30). This output is subsequently passed through an Activation Layer, producing the final Generated_Data of shape (30). The NS GAN Discriminator architecture displayed in Figure 8 (f) comprises various interconnected layers responsible for evaluating the authenticity of the generated data. This begins with an Input Layer, which takes an input shape of (1) and produces the same output shape. An Embedding Layer then processes the input to generate an output shape of (1, 30), followed

by flattening to result in an output shape of (30). Simultaneously, a Noise_Input Layer takes input of shape (30) and maintains the same output shape. The output of the Noise_Input Layer is concatenated with the flattened output from the Embedding Layer through a Concatenate Layer, generating an output shape of (60). This concatenated output is fed into a Dense Layer, producing an output shape of (128), followed by a Dropout Layer. Another Dense Layer follows, processing the output of the previous Dropout Layer to yield an output shape of (64), followed by another Dropout Layer. Lastly, another Dense Layer processes the output to produce an output shape of (32), which is then subjected to a final Dropout Layer. The Discriminator_out layer consists of a Dense Layer that takes input of shape (32) and yields an output shape of (1).

In the following experimentation we've implemented the LS GAN (Mao et al., 2017) method to oversample the minority class transactions. The architecture of the LSGAN Generator (depicted in Figure 8 (g)) involves a series of interconnected layers responsible for generating data through input manipulation. This begins with an Input Layer, accepting a single variable of shape (1) and producing an identical output shape. Subsequently, an Embedding Layer takes the input and generates an output shape of (1, 50), which is then flattened to (50). An additional input layer is included for noise, which maintains its input shape of (50). The output from this noise input layer is then concatenated with the flattened output from the previous layer, resulting in an output shape of (100). Two Dense Layers follow, with input shapes of (100) and (64) respectively, each accompanied by a Dropout Layer to maintain the output shape. A final Dense Layer follows, processing input of shape (64) and producing an output shape of (32), followed by a Dropout Layer. Ultimately, an Activation Layer processes the output, resulting in the generated data of shape (30). The LS GAN Discriminator architecture displayed in Figure 8 (h) encompasses a sequence of interconnected layers aimed at distinguishing between generated and real samples. Starting with an input layer, it takes a label of dimensions (1) and produces an output of the same shape. An Embedding Layer then processes the input to produce an output shape of (1, 30), followed by a Flatten Layer to yield an output of shape (30). Concurrently, a Noise_Input Layer processes input of shape (30) and maintains the same output shape. The outputs of both the Noise_Input Layer and the Flatten Layer are combined through a Concatenate Layer, resulting in an output shape of (60). A Dense Layer follows, with input shape (60) and output shape (128), along with a Dropout Layer maintaining the output shape. Another Dense Layer processes the output of the previous Dropout Layer, resulting in an output shape of (64), followed by another Dropout Layer. Finally, a further Dense Layer processes the output, generating an output shape of (32), succeeded by a final Dropout Layer. The Discriminator_out layer consists of a Dense Layer that processes input of shape (32) and yields an output of shape (1).

In our conducted experimentation, we implemented the K-CGAN method to oversample minority class transaction. The architecture of the K-CGAN Discriminator (illustrated in Figure 8 (j)) encompasses various interconnected layers essential for data processing and classification. The Input_label Input Layer serves as the initial layer, accepting input of shape (None, 1), signifying an unspecified batch size and a single neuron representing the condition vector. The subsequent Embedding Layer processes this single neuron, generating an embedding. This is followed by the Flatten Layer, which converts the input to a vector, succeeded by the Data_input layer. The Concatenate Layer follows, combining the flattened input with the data input. A Dense Layer creates feature maps, while the BatchNormalization layer ensures weight distribution normalisation. The LeakyRelu layer introduces non-linearity, and the Dropout layer randomly drops features to curb overfitting. Subsequently, more feature maps are generated by another Dense Layer, which is again normalized by the batch_normalization_1 layer. The LeakyRelu layer is reapplied, and the final feature maps are created by the Dense_5 layer, which then feeds into the Discriminator_out layer responsible for binary classification of real and generated data. The architecture of the GAN Discriminator ensures accurate classification and upholds desired standards for generated data. The K-CGAN Generator architecture displayed in Figure 8 (k) involves a sequence of interconnected layers collaborating to generate the required output. Commencing with the Input_label Input Layer, a conditional layer that takes a batch size and a single

neuron for the condition vector, it's succeeded by the Embedding Layer, which maps the input vector into a higher-dimensional space, providing an output for the subsequent layers. The Flatten Layer transforms the matrix into a vector, preparing it for further processing. Noise_input generates random noise data, which is concatenated with the previous layer's output through the Concatenate function. The following Dense layer undertakes a linear operation on the data, followed by the BatchNormalization layer. The tf.nn.reluTFOpLambda layer applies the ReLU activation function for introducing non-linearity in the generated data. To prevent overfitting, the Dropout layer follows, succeeded by another Dense layer, which also undergoes batch normalization. The tf.nn.relu_1 TFOpLambda layer applies ReLU once more, and the Dense_2 layer performs a linear operation on the data, producing the final generated data. This output is then returned through the Generated_Data layer. All these layers collaborate to generate realistic data closely resembling the original input.

4.3.6 GAN Training: Generator and Discriminator Losses

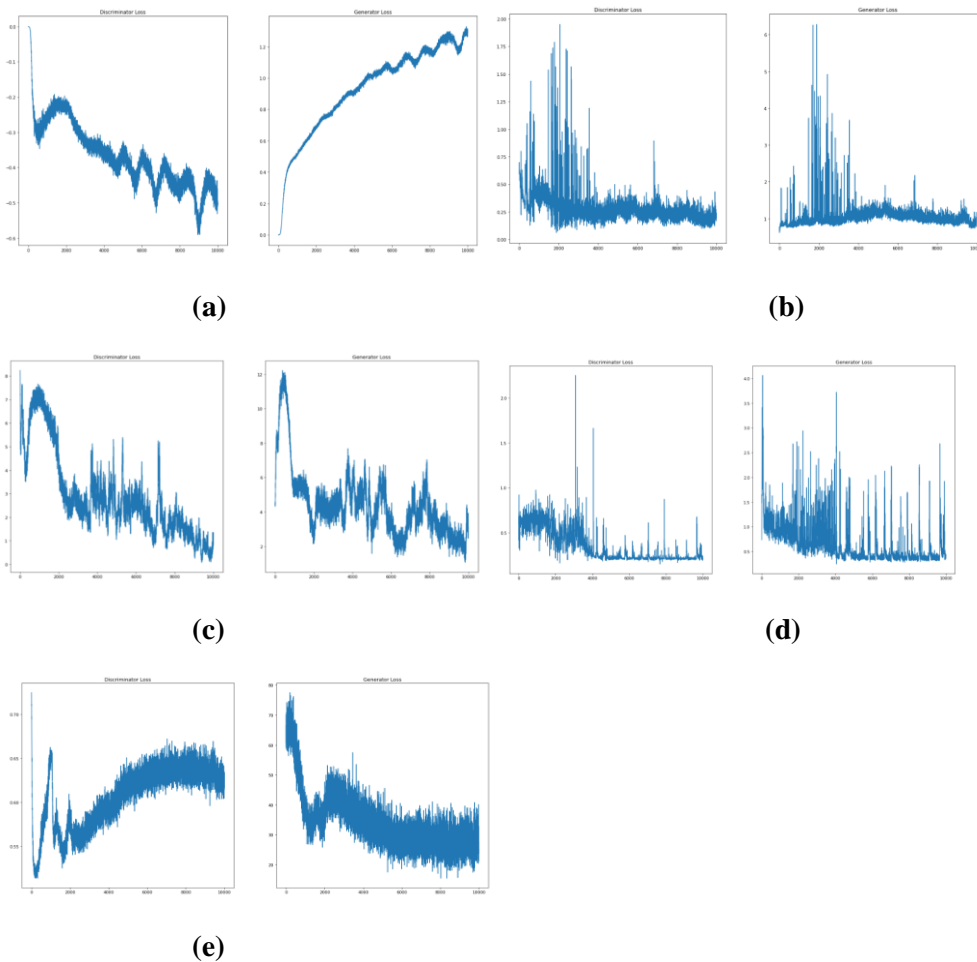


Figure 9: (a) The discriminator and generator losses of WGAN, (b) the discriminator and generator losses of SDG GAN (b), (c) the discriminator and generator losses of LS GAN, (d) the discriminator and generator losses of NS GAN, (e) the discriminator and generator losses of K-CGAN

WGAN

Figure 9 (a) presents the discriminator and generator losses of WGAN. The discriminator loss presented shows a significant decrease in the discriminator loss which indicates a significant improvement in its ability to distinguish between genuine and synthetic samples. This progression highlights considerable progress in the training process. However, contrary to this positive trend, generator losses reveal that the generator is having difficulty fabricating samples realistic enough to deceive the discriminator (Cao et al. 2019; Pfenninger et al., 2021). Possible reasons for this may include a deficiency in the design of the generator, which renders it unable to fabricate genuinely realistic samples (Wang et al., 2017). Alternatively, training instability (Wolterink et al., 2020) could also be a contributing factor to the underperformance of the generator. It is worth noting that mode collapse may also be a significant factor impacting the performance of the generator (Bhagyashree et al., 2020). This is characterised by the frequent generation of similar samples, which seem easy to create but do not adequately represent the entire distribution of actual data points (Ding et al., 2022; Zhang et al., 2018).

SDG GAN

As the analysis progressed, Figure 9 (b) demonstrates that it became apparent that the SDG GAN model had experienced a mode collapse after undergoing numerous epochs. After several epochs, a sharp upsurge in losses suggests that mode collapse occurred in the model. This phenomenon occurs when the generator generates only a limited subset of samples that are uncomplicated to produce but do not accurately represent the complete range of the true distribution. As a result, the discriminator becomes exceptionally efficient at distinguishing between real and synthetic samples, leading to a sudden spike in losses. Furthermore, the model displayed signs of convergence issues, as evidenced by the abrupt increase in losses followed by erratic fluctuations. These fluctuations indicate that the model was switching between various modes, suggesting that it was not able to maintain a stable and consistent learning process (Charitou et al., 2021; Figueira and Vaz, 2022). Overall, the findings suggest that mode collapse and convergence issues significantly impact the effectiveness and accuracy of this generative model. These issues must be carefully monitored and addressed to ensure that the model can produce high-quality and natural-looking samples that accurately reflect the true distribution (Saxena and Cao, 2021). Further research is required to develop robust solutions for these problems and enhance the overall performance of generative models.

LS GAN

Figure 9 (c) shows inconsistent losses for both the generator and discriminator of the LS GAN model. This indicates unsteady training, which in turn signifies that the model is struggling to achieve convergence (Wang et al., 2019; Zhong et al., 2022). The issue of non-convergence is a significant challenge in GAN training and can have a detrimental effect on the model's output quality. Multiple factors could contribute to this phenomenon, including poor hyperparameter selection, an unstable architecture or mode collapse. It is imperative to mention that hyperparameter selection is critical for GANs, as it directly affects the loss function's behaviour during training (Alarsan and Younes, 2021; Fiore et al., 2019). An unoptimized loss function could lead to unsteady training, causing the model to oscillate erratically. Furthermore, an unstable architecture could be a factor. The design of the generator and discriminator could result in gradients that are too large or small, leading to poor convergence. Mode collapse, another possible explanation, could occur when the generator learns to generate data that is similar to a small subset of the training data. In this regard, the LS GAN was unable to achieve desired results.

NS GAN

Figure 9 (d) shows the losses of NS GAN. After several training cycles, the observed sudden spikes in losses strongly indicate the presence of a probable mode collapse (Hong et al., 2019). Specifically, it appears that the generator is only capable of producing a narrow range of sample types that are relatively simple to generate and do not accurately reflect the true distribution of data. As a result, the discriminator may become exceedingly adept at differentiating between real and synthetic samples, leading to a sharp increase in overall

model losses. This phenomenon is further supported by regular dips in the loss function after a sudden surge, indicating that the model struggles to maintain a consistent performance level and instead periodically oscillates between disparate states. These fluctuations can greatly impede the accurate modelling of complex real-world data distributions and highlight the need for more effective model design and training strategies (Chen, 2021; Pan et al., 2019).

Novelty Loss K-CGAN

Figure 9 (e) shows the discriminator and generator losses of Novelty K-CGAN. The discriminator loss initially decreasing followed by an increase implies that the discriminator is effectively distinguishing between real and generated samples, however it starts to struggle as the generator produces more realistic samples. The discriminator loss may have been amplified due to the generator constructing samples that are more intricate and indistinguishable from an authentic data distribution. It appears that the generator is having trouble learning a dependable representation of the data distribution, evidenced by its wavering performance. This could be caused by either a complexity of the data distribution or noisy input dataset (Chiaroni et al., 2019; Pavan et al., 2021).

4.3.7 Results and Comparisons of Classification Models

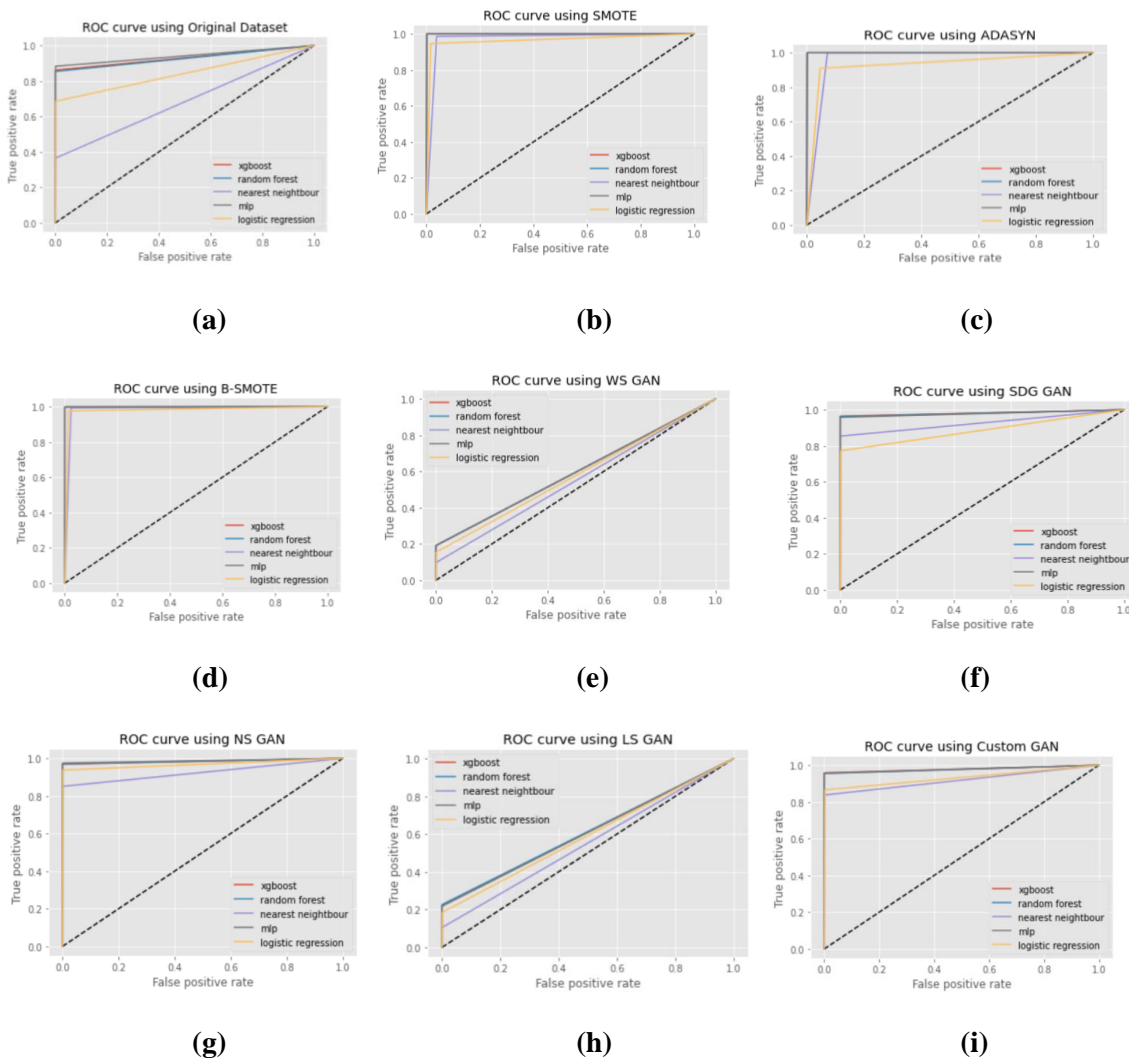


Figure 10: (a) ROC curve for original imbalanced dataset, (b) ROC curve for balanced dataset utilising SMOTE, (c) ROC curve for balanced dataset utilising ADASYN, (d) ROC curve for balanced dataset utilising B-SMOTE, (e) ROC curve for balanced dataset utilising WGAN, (f) ROC curve for balanced dataset

utilising SDG GAN, (g) ROC curve for balanced dataset utilising NS GAN, (h) ROC curve for balanced dataset utilising LS GAN, (i) ROC curve for balanced dataset utilising K-CGAN

Figure 10 presents a series of ROC curves showcasing the performance of different techniques on a balanced dataset. The curves represent the results obtained using various approaches, namely SMOTE, ADASYN, B-SMOTE, WGAN, SDG GAN, NS GAN, LS GAN, and K-CGAN. Each curve represents the performance of the respective technique in achieving balance in the dataset. These ROC curves provide valuable insights into the effectiveness of different methods in addressing the issue of imbalance in the dataset. Table 9 presents the evaluation of the classifiers applied to the original imbalanced dataset. The findings indicate that XGBoost demonstrated impressive performance, while Random Forest, Nearest Neighbour, MLP and Logistic Regression have displayed satisfactory performance levels. The results obtained could be used to select the most appropriate classifier in future applications of this nature. The performance of five popular classifiers on the original imbalanced dataset has been analysed. Amongst the classifiers, XGBoost has exhibited the highest precision of 0.9916, with a satisfactory recall score of 0.8613. Hence, achieving the highest F1 score of 0.9219 and accuracy of 0.9984, surpassing all other classifiers evaluated in this study. The Random Forest classifier has demonstrated a precision score of 0.9512 and a recall score of 0.8540. Consequently, obtaining an F1 score of 0.9000 and an accuracy score of 0.9979, representing satisfactory scores, placed it in the third position based on the classifier's performance metrics. On the other hand, the Nearest Neighbor classifier has shown a high precision score of 0.9259. However, its recall score of 0.3649 has adversely affected its F1 score of 0.5236. Furthermore, the MLP also placed in the second position after XGBoost. The MLP classifier has obtained a precision score of 0.9308 and a recall score of 0.8832, achieving an F1 score of 0.9064, and an accuracy score of 0.9980. While the Logistic Regression classifier has achieved a precision score of 0.8785 and a recall score of 0.6861, acquiring an F1 score of 0.7705, and an accuracy score of 0.9955.

Table 9: Classifiers performance on original imbalanced dataset

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.9916	0.8613	0.9219	0.9984
Random Forest	0.9512	0.8540	0.9000	0.9979
Nearest Neighbor	0.9259	0.3649	0.5236	0.9927
MLP	0.9308	0.8832	0.9064	0.9980
Logistic Regression	0.8785	0.6861	0.7705	0.9955

Table 10 shows the performance of five classifiers on a balanced dataset that was oversampled using SMOTE to address the issue of imbalanced data. Precision, recall, F1 score, and accuracy were the key metrics used to evaluate the performance of the classifiers. Based on our findings shown in Table 10 its evident that XGBoost and Random Forest classifiers exhibited high performance in all the metrics using SMOTE, with F1 scores of 0.9994 and 0.9995, respectively. These algorithms are known to be robust and efficient in handling imbalanced datasets and are widely used in various applications, such as fraud detection and intrusion detection. On the other hand, the Nearest Neighbor and Logistic Regression classifiers showed a relatively lower performance in terms of precision, recall, and F1 score and accuracy. The MLP classifier also performed well with an F1 score of 0.9992. Finally, Logistic Regression exhibited a decent performance, except for the recall value which was 0.9465 and considerably lower than the other classifiers. This phenomenon can be attributed to the classifier's assumptions, and it is recommended to use it only for specific

scenarios where these assumptions are met. The performance of the classifiers on the balanced dataset using the SMOTE oversampling technique shows that the XGBoost and Random Forest classifiers are the most reliable options. However, the suitability of the classifiers may vary based on the specific use case, and further analysis is recommended.

Table 10: Classifiers performance on balanced dataset SMOTE oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.9989	0.9998	0.9994	0.9994
Random Forest	0.9993	0.9998	0.9995	0.9995
Nearest Neighbor	0.9634	0.9865	0.9748	0.9743
MLP	0.9986	0.9998	0.9992	0.9992
Logistic Regression	0.9836	0.9465	0.9647	0.9651

Table 11 shows the performance of five classifiers on a balanced dataset that was oversampled using ADASYN to address the issue of imbalanced data. Precision, recall, F1 score and accuracy were the key metrics used to evaluate the performance of the classifiers.

Table 11: Classifiers performance on balanced dataset ADASYN oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.9989	1.0000	0.9995	0.9995
Random Forest	0.9995	1.0000	0.9998	0.9998
Nearest Neighbor	0.9310	0.9986	0.9636	0.9627
MLP	0.9982	0.9999	0.9991	0.9991
Logistic Regression	0.9501	0.9098	0.9295	0.9318

Table 11 presents classifiers performance on a balanced dataset upon application of ADASYN method oversampling the minority class. Based on the findings, XGBoost shows high performance results in terms of precision, recall and accuracy for detecting the minority class, with a near-perfect F1 score of 0.9995. This indicates that XGBoost is highly effective at identifying both positive and negative instances. Moreover, Random Forest performed well with the highest scores with almost perfect precision, recall, and F1 score values, the model achieved a very high level of accuracy. Further, Nearest Neighbor demonstrated a slightly lower level of precision and F1 score, which suggests that the model struggles with detecting negative instances. However, the model's recall value of 0.9986 indicates its effectiveness at correctly identifying positive instances, which presents the usefulness of this classifier in specific contexts. MLP also showed high results, particularly with its high recall score, which means that it is successful in identifying as many True Positive (TP) instances as possible. The model's precision, accuracy, and F1 score values were also noteworthy, highlighting the overall potency of MLP. Lastly, the performance of Logistic Regression was slightly lower than that of the other classifiers, with a lower precision, recall values and F1 score.

Table 12: Classifiers performance on balanced dataset B-SMOTE oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.999757	0.999111	0.999434	0.999434
Random Forest	0.999919	0.999111	0.999515	0.999515
Nearest Neighbor	0.976018	0.993052	0.984461	0.984326
MLP	0.999514	0.997172	0.998342	0.998344
Logistic Regression	0.985890	0.976650	0.981248	0.981336

Table 12 shows the performance of five classifiers on a balanced dataset that was oversampled using B-SMOTE to address the issue of imbalanced data. The results shown in Table 12 reveal that XGBoost and Random Forest exhibit superior performance in terms of precision, recall, and F1 score, with the F1 scores of 0.999434 and 0.999515. The high F1 scores imply that these classifiers have a superior balance between precision and recall, indicating their robustness in identifying TP cases, while minimising the number of FPs and False Negatives (FNs). However, the performance of the Nearest Neighbor classifier was slightly lower than XGBoost and Random Forest, with an F1 score of 0.984461. This suggests that the Nearest Neighbor classifier struggles to find minority class samples and may have difficulty classifying highly imbalanced datasets. On the other hand, the MLP and Logistic Regression classifiers also demonstrate slightly lower performance than XGBoost and Random Forest. However, the scores are still impressive as both the classifiers managed to achieve F1 scores of 0.998342 and 0.981248. The high F1 score of the MLP classifier indicates its ability to correctly identify minority class instances while maintaining a low number of false positives and FNs. Similarly, the Logistic Regression classifier effectively predicted the minority class instances, but was slightly inferior in terms of F1 score compared to other classifiers. The results suggest that XGBoost and Random Forest classifiers are the most suitable for data classification using B-SMOTE oversampling technique while MLP and Logistic Regression are also strong performers. Nonetheless, the choice of classifier will depend on many factors, such as the nature of the data, the available computation resources, and the specific requirements of the classification task.

For SDG GAN, as shown in Table 13, XGBoost and Random Forest perform the best in terms of precision, recall, F1 score, and accuracy. Nearest Neighbour performs the worst in terms of recall and F1 score, but still has a high accuracy. Logistic Regression has the lowest precision, recall, and F1 score, but still has a relatively high accuracy.

Table 13: Classifiers performance on balanced dataset using SDG GAN by oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.997840	0.964509	0.980892	0.999287
Random Forest	0.995680	0.962422	0.978769	0.999208
Nearest Neighbor	0.997561	0.853862	0.920135	0.997189
MLP	0.989224	0.958246	0.973489	0.999010
Logistic Regression	0.906863	0.772443	0.834273	0.994179

The results in Table 13 demonstrate the performance of five classifiers on a balanced dataset using SDG GAN oversampling the minority class. Based on the values of each metrics, it's evident that XGBoost and Random Forest performed the best with precision, recall, F1 score, and accuracy values. XGBoost achieved precision, recall, F1 score, and accuracy values of 0.997840, 0.964509, 0.980892 and 0.999287. On the other hand, Random Forest managed to achieve precision, recall, F1 score, and accuracy values of 0.995680, 0.962422, 0.978769 and 0.999208. These high scores indicate that SDG GAN can effectively distinguish between the majority and minority classes. Nearest Neighbour underperformed XGBoost and Random Forest in terms of recall and F1 score, but still achieved a high accuracy value of 0.997189. Logistic Regression had the lowest precision, recall, and F1 score values, but still achieved an accuracy of 0.994179. These results suggest that XGBoost and Random Forest have the highest variable similarity to balanced data with SDG GAN oversampling. MLP also demonstrated good performance, indicating that it can effectively differentiate between classes. Furthermore, these results demonstrate that with SDG GAN, Logistic Regression can still reach a relatively high accuracy value, even with lower precision, recall, and F1 scores.

Overall, the results in suggest that SDG GAN is an effective oversampling technique for imbalanced datasets. It allows for high accuracy values while still maintaining variable similarity amongst the different classifiers. This makes it a powerful tool for dealing with imbalanced datasets where the minority class has to be accurately identified.

Table 14: Classifiers performance on imbalanced dataset NS GAN oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	1.000	0.969	0.984	0.999
Random Forest	0.996	0.971	0.983	0.999
Nearest Neighbor	0.995	0.851	0.918	0.997
MLP	0.979	0.975	0.977	0.999
Logistic Regression	0.948	0.938	0.943	0.998

Table 14 shows the findings of oversampling the minority class using NS GAN. The results show that XGBoost and Random Forest models achieved the highest precision rates of 1.000 and 0.996. At the same time, the Nearest Neighbor model did not fall far behind with a precision of 0.995. The MLP model, on the other hand, demonstrated a relatively lower precision rate of 0.979. The Logistic Regression model obtained the lowest precision result of 0.948. In terms of recall, Random Forest and MLP models displayed near-perfect rates of 0.971 and 0.975 respectively. The Logistic Regression and Nearest Neighbor models performed relatively lower with a recall rate of 0.948 and 0.851. The XGBoost model achieved a reasonable recall rate of 0.969. The F1 score, a measure impacting both precision and recall, showed that the XGBoost and Random Forest models achieved the highest rates of 0.984 and 0.983. The Logistic Regression and Nearest Neighbor models performed relatively lower with an F1 score of 0.943 and 0.918, while the MLP model demonstrated a decent F1 score of 0.977. Finally, when considering accuracy, all models achieved high rates, with the MLP, XGBoost and Random Forest classifiers obtaining the highest accuracies of 0.999. The Nearest Neighbor and Logistic Regression models achieved the accuracy rates of 0.997 and 0.998. Overall, the Table 14 reflecting classifiers' performance on the balanced dataset using NS GAN in order to oversample the minority class presents a promising range of models that can be utilized to achieve superior outcomes in data analysis and prediction tasks, particularly in cases where the minority class is underrepresented.

Table 15 shows the performance of five classifiers on a balanced dataset that was oversampled using LS GAN model to address the issue of imbalanced data. Precision, recall, F1 score, and accuracy were the key metrics used to evaluate the performance of the classifiers.

Table 15: Classifiers performance on imbalanced dataset using LS GAN to oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.980583	0.220044	0.359431	0.985745
Random Forest	0.990476	0.226580	0.368794	0.985903
Nearest Neighbor	0.979592	0.104575	0.188976	0.983686
MLP	0.952381	0.217865	0.354610	0.985586
Logistic Regression	0.850000	0.185185	0.304114	0.984596

Table 15 shows classifiers performance using LS GAN to oversampling the minority class, where XGBoost and Random Forest yielded the highest precision scores of 0.980583 and 0.990476, respectively. This metric indicates the ratio of correctly predicted positive observations to the total predicted positive observations, highlighting the classifier's ability to avoid false positives. However, the recall scores obtained for all classifiers were relatively low, rarely exceeding 0.22. Recall measures the ratio of correctly predicted positive observations to the total actual positive observations and brings attention to the classifier's ability to detect positive samples, in this case, the minority class. The low recall scores indicate that all classifiers had difficulty correctly detecting positive samples. The F1 scores are a harmonic mean of precision and recall, giving equal weight to both metrics. It can be observed that the F1 scores were relatively low, ranging from 0.188976 to 0.368794. This indicates that the classifiers' performance on accurately classifying the minority class is suboptimal. Finally, the accuracy score, which measures the proportion of the total number of predictions that are correct, showed high results for all classifiers, with scores ranging from 0.983686 to 0.985903. However, accuracy can be misleading where the classifier can achieve high accuracy by simply predicting the majority class frequently. Overall, while the XGBoost and Random Forest classifiers showed

high precision scores, all classifiers had low recall and F1 scores on the LS GAN oversampling minority class dataset. These results suggest that the classifiers have difficulty detecting the minority class. The results pointed out that Random Forest and XGBoost have outperformed all other examined models in terms of remarkable reliability for the LS GAN approach.

Furthermore, Table 16 shows the performance of five classifiers on a balanced dataset that was oversampled using K-CGAN to address the issue of imbalanced data. Precision, recall, F1 score, and accuracy were the key metrics used to evaluate the performance of the classifiers

Table 16: Classifiers performance on imbalanced dataset using Novelty Loss K-CGAN by oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.998039	0.958569	0.977906	0.999089
Random Forest	0.988304	0.954802	0.971264	0.998812
Nearest Neighbor	0.991091	0.838041	0.908163	0.996436
MLP	0.980695	0.956685	0.968541	0.998693
Logistic Regression	0.87619	0.86629	0.871212	0.994615

The performance of Novelty Loss K-CGAN in addressing imbalanced datasets is analyzed using the results from Table 16. The table presents the performance of five different classifiers, namely XGBoost, Random Forest, Nearest Neighbor, MLP, and Logistic Regression, on an imbalanced dataset. The dataset was first oversampled using Novelty Loss K-CGAN before being tested with the classifiers. Based on the results presented in Table 16, all the classifiers obtained high values of Precision and F1 Score except Logistic Regression. This indicates that the models were able to accurately classify positive instances, despite the imbalanced nature of the dataset. XGBoost achieved the highest Precision value of 0.998039, followed closely by Nearest Neighbor with 0.991091. Random Forest obtained the third-highest value of 0.988304, indicating its ability to handle imbalanced datasets. In terms of Recall, the results show that the Nearest Neighbor and Logistic Regression classifiers achieved the lowest value of 0.838041 and 0.86629. This suggests that the model struggled to identify all the positive instances in the dataset. However, all the other classifiers achieved Recall values above 0.95, meaning they could accurately identify most of the positive instances in the dataset. In terms of Accuracy, all the classifiers obtained high scores, ranging from 0.994615 to 0.999089. This further confirms that the models were able to accurately classify most of the instances in the dataset. Overall, the results suggest that Novelty Loss K-CGAN can be an effective technique for addressing imbalanced datasets. All the classifiers were able to achieve high values of Precision, F1 Score, and Accuracy, indicating their ability to accurately classify positive instances. The results further suggest that MLP, XGBoost, and Random Forest classifiers can be effective with K-CGAN. However, it is noteworthy to mention that the choice of classifier may depend on specific use cases and the nature of the datasets being analyzed. The findings pointed out that Precision, recall, F1 score, and accuracy were all best enhanced by using XGBoost and Random Forest models. Logistic Regression had the least accuracy compared to the other models. However, K-CGAN approach integration overall improved performance of classification.

Discussion and Conclusions

There could be several reasons why the classifiers' performance of Novelty Custom K-CGAN is not achieving the same level of success as other GAN models on the same dataset. One possible explanation is that while the KL loss is a widely accepted measure for distribution similarity, it may overlook certain characteristics or patterns in the data, leading to mediocre performance from novelty GAN models. Thus, this dataset may require further examination to ensure that no important elements go unnoticed and unexpected outcomes are avoided.

Another explanation is that these novelty GAN models have not been specifically optimised in experiment for the dataset, which could lead to less consistent performance across different experiments. Although the hyperparameters of the Novelty Custom K-CGAN may not have been calibrated to best suit every aspect of the dataset, its potency is still undeniable. Furthermore, if GAN models are not initialized with consistent, optimal weights, this can cause deviations in performance from one experiment to the next. Poor initialization of GAN model weights means that novelty models may suffer and risk producing unreliable results. Therefore, it is important to carefully consider the initialization process to ensure optimal performance. The results show that the precision, recall, F1 score, and accuracy metrics can vary significantly between various classification models and various GAN-based methods. Overall, the performance of the XGBoost and Random forest classification models outperforms all other models (Faraji, 2022).

The performance of NS GAN and SDG GAN is superior to all previous iterations of the Generative Adversarial Network (GAN) in terms of accuracy, recall, F1 score, and precision. For every classification model, the LS GAN performs the worst among all GANs. While the service is not consistent across all models, K-CGAN show greater performance for some models in comparison to other types of GANs (Fiore et al., 2019). In the next set of experiments we're going to further develop and optimise the architecture of K-CGAN and other GANs.

4.4 Development of K-CGAN Framework: Impact of Custom Loss

In the following experiments we aimed to evaluate the impact of KL divergence on performance of the GAN model as well as assess the impact of adding SMOTE oversampling the minority class instances of fraudulent transactions to ensure sufficient representation for both classes in the dataset used specifically for GAN training. We have examined the Cosine similarity scores, the cosine similarity measures the similarity between the synthetic and original datasets based on their feature vectors (Büttcher et al., 2010), as demonstrated in Figure 11.

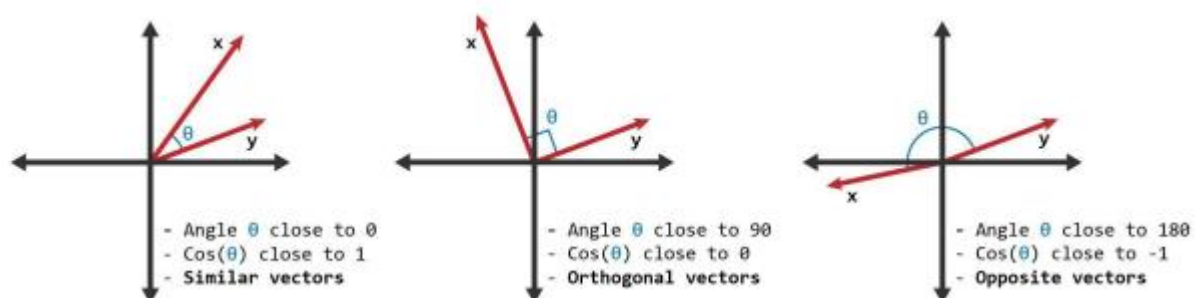


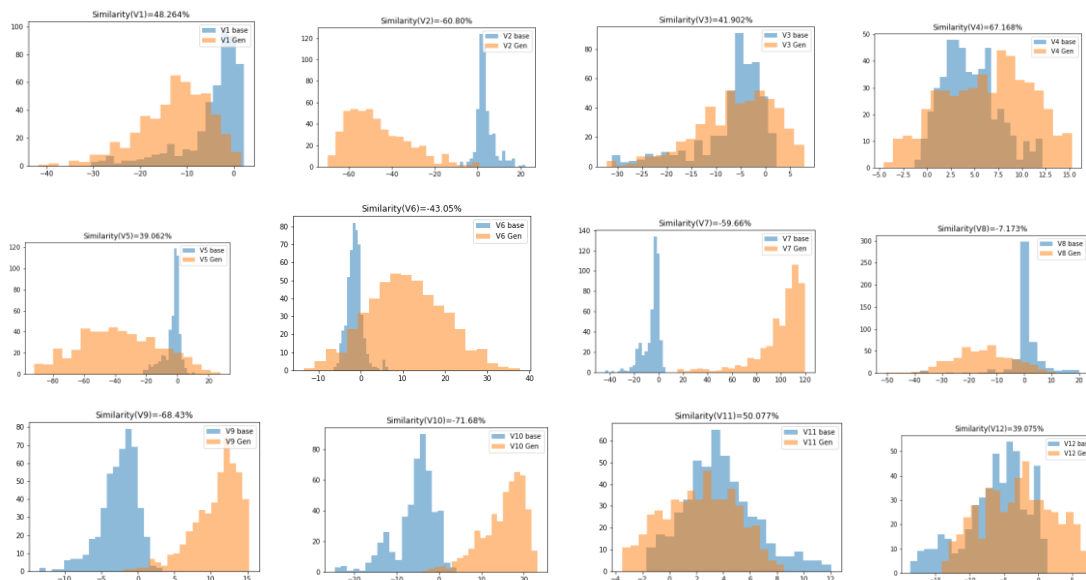
Figure 11: Cosine Similarity strategy

In our original credit card dataset we have V1 to V28 and Amount features so total 29 features we want to predict using the K-CGAN model. So if $y = V1$ (taken from Original dataset) and $x = V1$ (taken from Generated dataset) then our goal is to have ZERO ANGLE between x and y. And $\text{COS}(0) = 1$, that's why we would also state that if cosine similarity is 1 then it's an optimal value. This ensures that the generated

features closely resemble the original ones, allowing for accurate predictions in credit card fraud detection. Cosine similarity is a measure of the similarity between two vectors in space, which in this case, are the different features of the dataset. A high cosine similarity score indicates that the two features are similar, while a low score indicates that they are dissimilar. Cosine similarity analysis is useful in machine learning and data mining to identify patterns and relationships within datasets. By understanding the similarity between features, it is possible to create models that accurately predict outcomes and detect anomalies.

4.4.1 Experiment 1: K-CGAN with Novelty KL Loss without SMOTE

Experiment 1 utilised novelty custom K-CGAN method without specific modifications training with 100 epochs. Figure 12 demonstrates the results of the experiment, showing that the K-CGAN method was able to generate fraudulent transactions with the cosine similarity scores ranging from -71.68% (V10) to 67.17% (V4). The blue coloured denotes to original feature and the orange to the generated feature data. In general, the cosine similarity values for V1-V17 were higher than those of V18-V28 and the 'Amount'. The highest similarity was observed for V4, followed by V14 and V11. The lowest cosine similarity scores were observed for the variables V9 and V10, indicating that these two variables were least similar between the original and synthetic datasets. This result is significant as it suggests that the K-CGAN was able to generate unseen fraudulent transactions with similar feature values to that of the original data. Furthermore, these results indicate that our proposed K-CGAN is an effective approach for generating synthetic fraudulent transactions with high accuracy. However, the cosine scores are still much lower than anticipated. To improve the similarity between the generated and original dataset, the study conducted more experiments in order to improve the accuracy of the novelty loss K-CGAN. It is important to note that the results of this experiment are limited due to the small size of the dataset used and other factors such as feature selection. The synthetic data generated was also not evaluated using a classifier, thus further experiments will focus on evaluating these two aspects in order to determine the GAN's true effectiveness. Furthermore, other techniques such as SMOTE should be considered during training (further presented in experiment 2) in order to assist with GAN training.



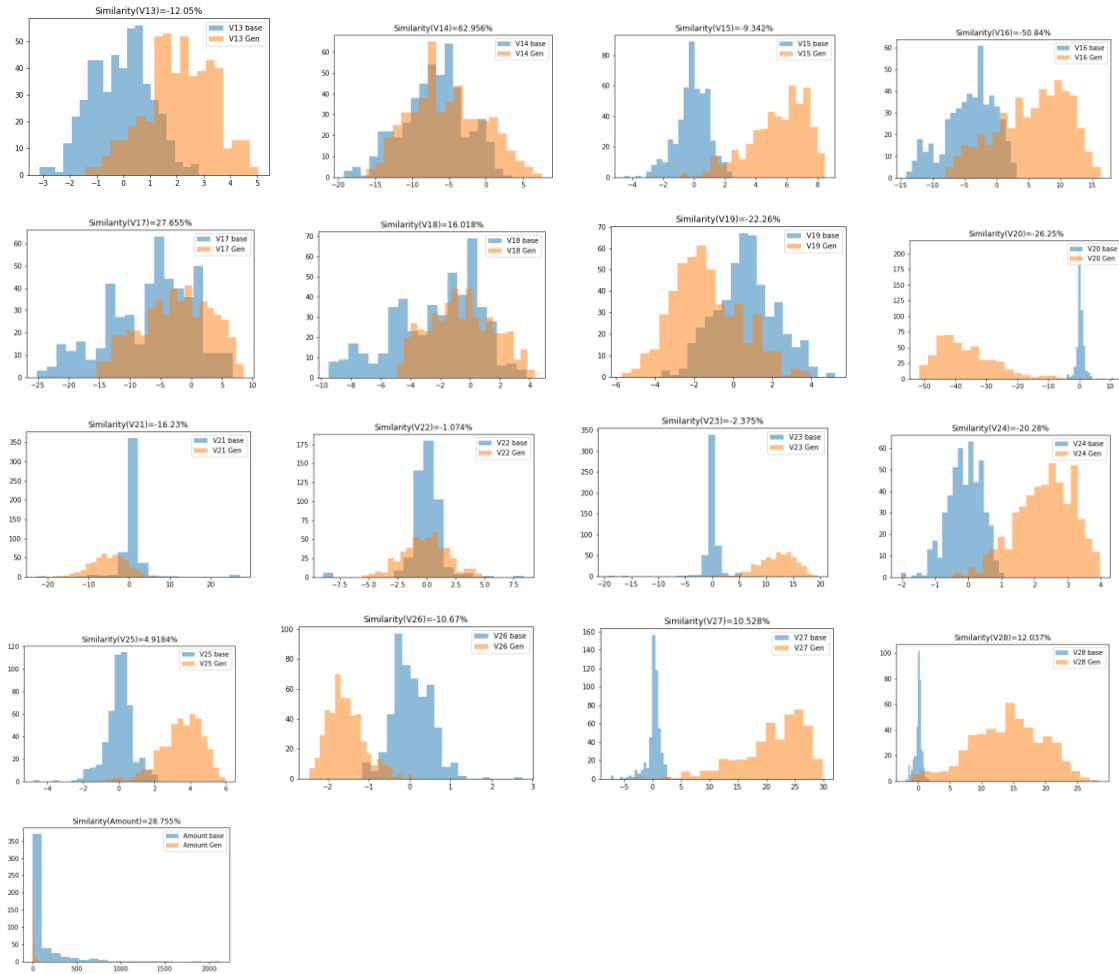
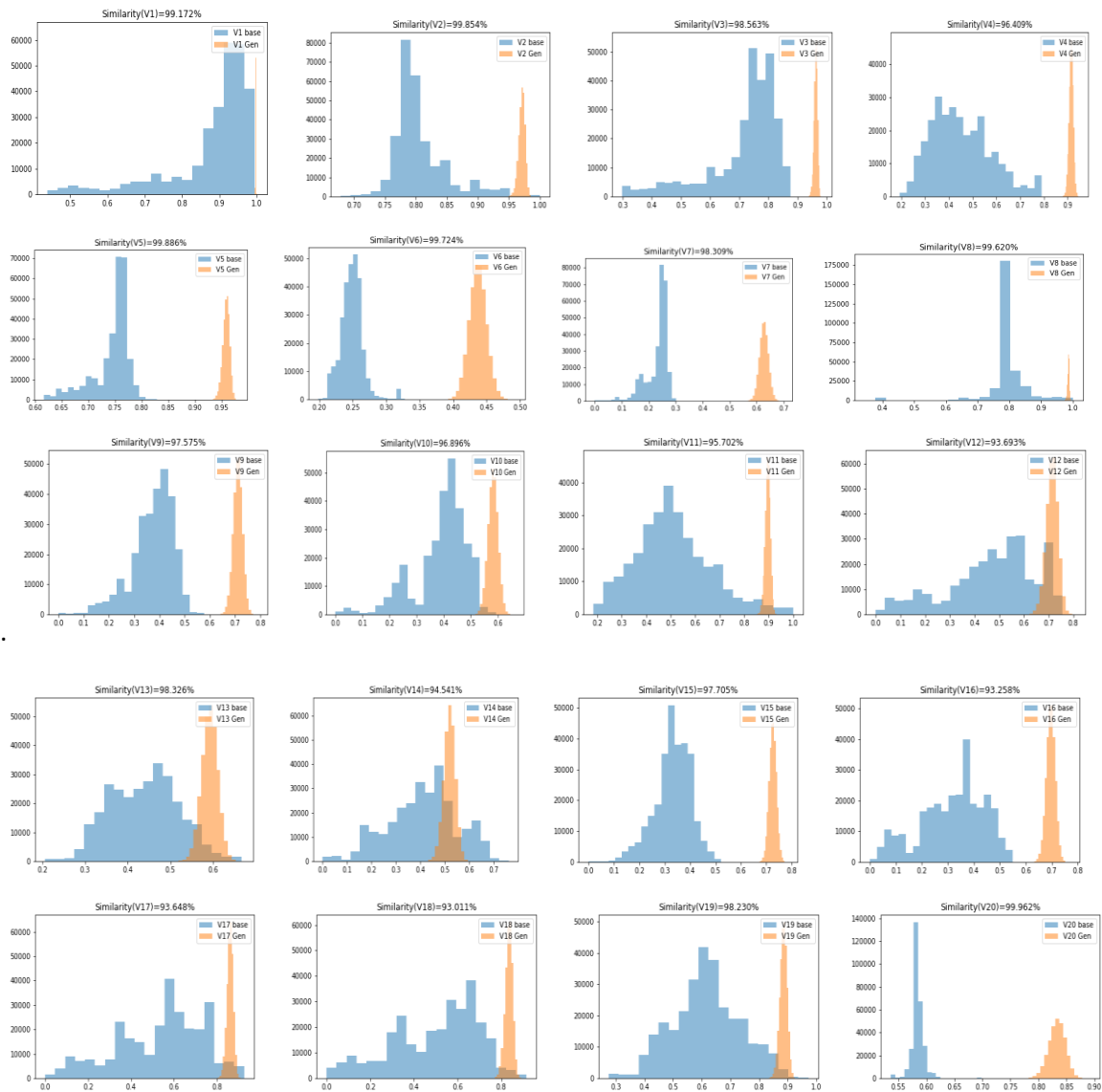


Figure 12: Distribution charts showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 for experiment 1

4.4.2 Experiment 2: K-CGAN with Novelty KL Loss with SMOTE

In experiment 2, we oversampled the minority class instances of fraudulent transactions using SMOTE to ensure sufficient representation for both classes in the dataset used for GAN training. This step was crucial because models trained on imbalanced datasets often exhibit poor generalisation performance when faced with unseen data (He and Wu, 2019). During the training phase, we utilised the entire balanced dataset to ensure comprehensive coverage of the data nuances. No additional preprocessing techniques were necessary as the dataset had already undergone meticulous cleaning and labelling. While the 80-20% split is typically used for testing purposes, we omitted this step as our intention was to compare results with a synthetic dataset. During training, SMOTE was applied to generate additional synthetic fraudulent samples, thereby balancing the training dataset. This approach ensured that the GAN learned to generate realistic fraudulent samples and avoided bias towards the majority class. By capturing the underlying distribution of both classes, the generator achieved improved performance and enhanced generalisation. It's important to highlight that once the GAN is trained, SMOTE is no longer required for generating synthetic samples. Furthermore in this experiment we have utilised our custom K-CGAN method where Generator's loss is KL divergence and Binary Cross entropy, while Discriminator's loss Binary Cross entropy.

During the generation phase, the trained generator can directly produce synthetic fraudulent transactions. Similarly, during the evaluation phase, when assessing the K-CGAN's performance on unseen or real-world data, SMOTE is unnecessary. We can utilise the trained discriminator to classify transactions as legitimate or fraudulent, and the trained generator to generate synthetic fraudulent samples for comparison or analysis. In summary, SMOTE was only employed during K-CGAN training to address imbalanced datasets, but it is not applied during the generation or evaluation phases once the K-CGAN is trained. For the testing phase, we utilised a sample size of 284,315 fraudulent transactions with the Novelty K-CGAN. The same sample size was used for the oversampled dataset to facilitate comparison against our model results. In this study, we used distribution chart analysis to compare the generated dataset with the original dataset. Figure 13 charts show the comparison of all features for the experiment 2.



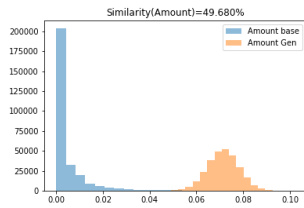
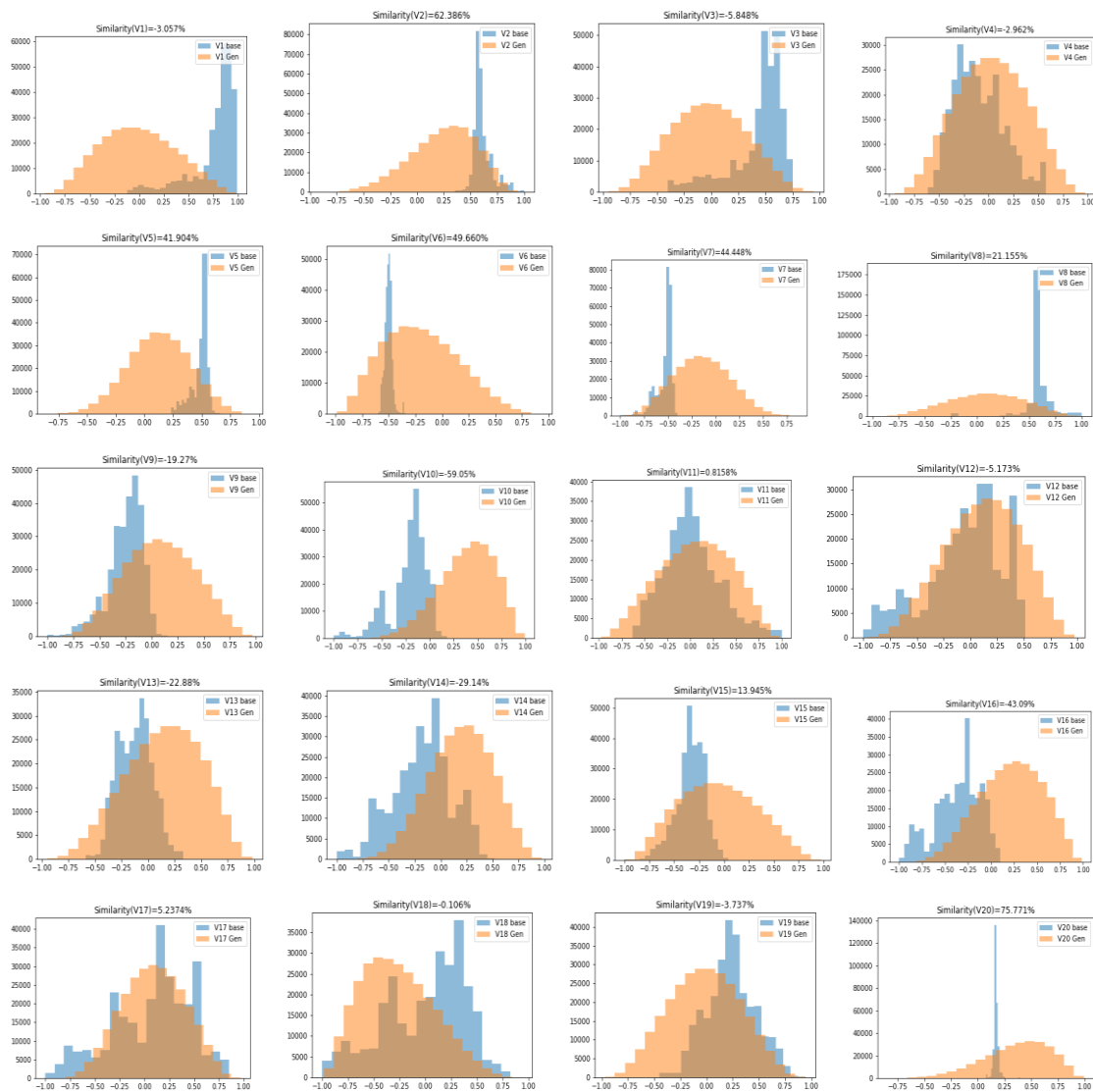


Figure 13: Distribution charts showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 for experiment 2 with implementation of SMOTE to assist with K-CGAN (with KL divergence) training

The distribution charts of Figure 13 suggest that the improved similarity scores of features V1 to V28 and Amount strongly imply the effectiveness of adopting SMOTE to assist with K-CGAN training. This improvement becomes even more pronounced when evaluating K-CGAN performance using the Kullback-Leibler Divergence (KL). Thus, it can be inferred that the use of KL loss K-CGAN with SMOTE oversampling serves as a valuable pre-training technique for generating synthetic data that closely approximates the characteristics of the original dataset.

4.4.3 Experiment 3: K-CGAN without KL Loss with SMOTE

For the experiment 3 we applied SMOTE to oversample the minority class instances of fraudulent transactions as in the experiment 2. Moreover, we utilised custom K-CGAN without KL Loss. The goal was to evaluate the impact of KL loss hence we simulated the same K-CGAN architecture where instead of KL loss, the study utilised standard Binary Cross Entropy (Goodfellow et al., 2016) loss function in both Generator and Discriminator networks. For the training phase, the entire dataset was used instead of the split. Similarly, for the testing phase, we conducted an experiment with 284,315 fraudulent transactions to evaluate the performance of our K-CGAN method. We compared this to the oversampled dataset which was also constructed with the same number of fraudulent transactions. In this study, we used distribution chart analysis to compare the generated dataset with the original dataset, the comparison of all features, representative of individual features, is demonstrated in Figure 14.



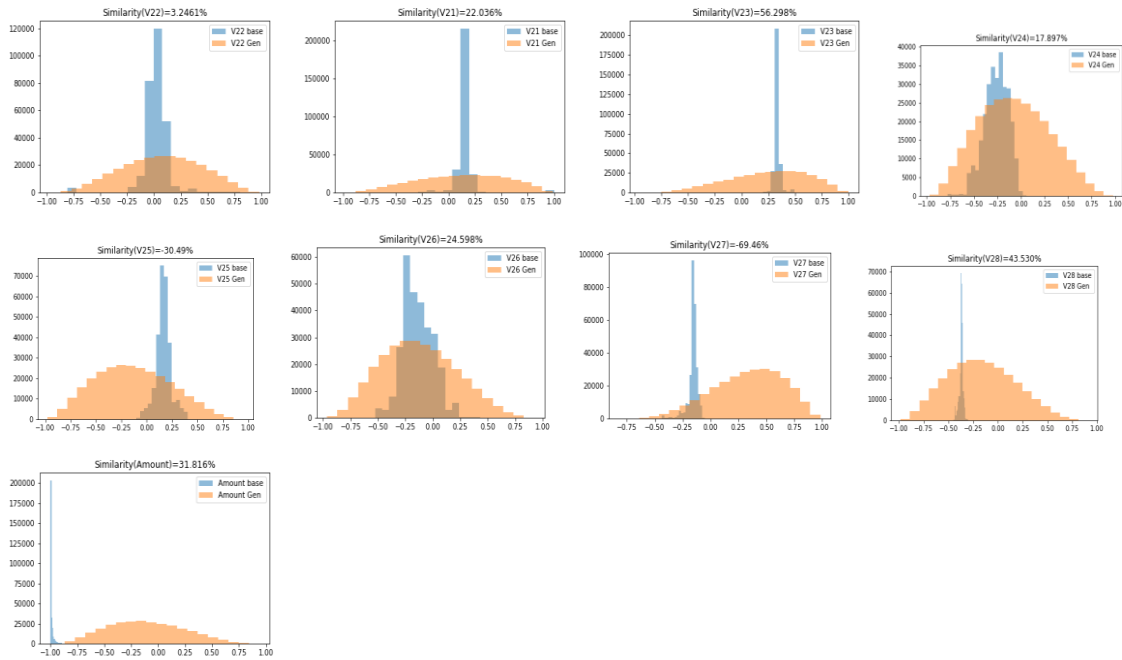
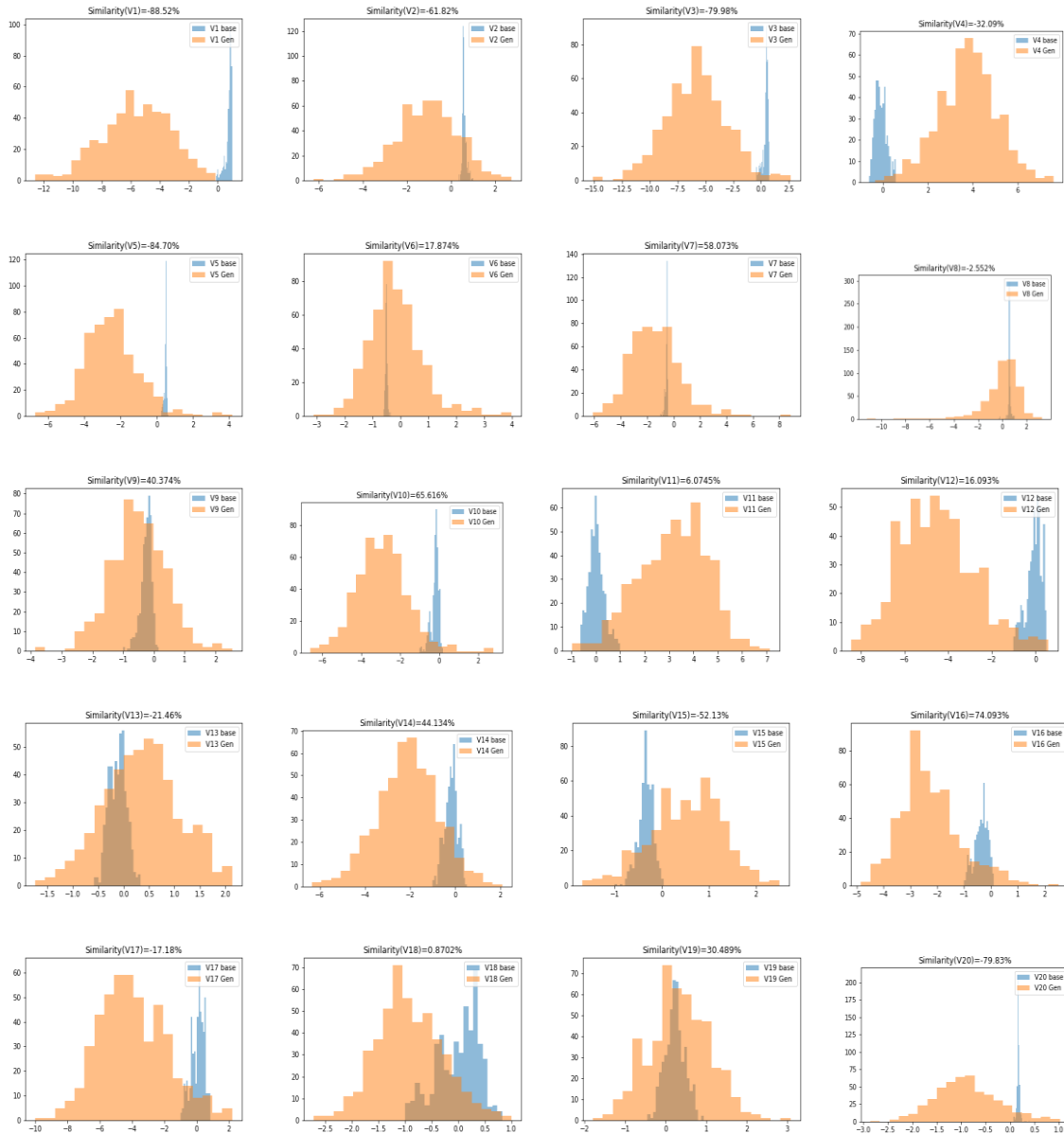


Figure 14: Distribution charts showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 experiment 3 without KL Loss GAN with SMOTE

Figure 14 distribution charts show the cosine similarity scores of all features. When compared to the baseline experiment from experiment 2, it can be seen that there is a noticeable reduction of similarity scores in this third experiment. As such, the elimination of KL Loss GAN with SMOTE has had an effect on reducing the cosine similarities between features and Amount. This suggests that adding KL loss to the GAN effectively increased its ability to improve the quality of the synthetic data generated by the K-CGAN model.

4.4.4 Experiment 4: K-CGAN with Novelty KL Loss without SMOTE

The purpose of our experimental study was to investigate the effectiveness of our newly developed novelty loss function where Generator's loss is KL divergence and Binary cross entropy and Discriminator's loss is Binary Cross entropy. In this experiment the study used the entire original dataset for training, as 80-20% split was not necessary since we planned to conduct a statistical comparison with the synthetic dataset. Moreover for the testing phase, the novelty K-CGAN was used to generate 492 fraudulent transactions for comparison with the oversampled dataset. We chose this number as it provided an equal proportion of fraud and legitimate samples in both datasets. In this study, we used distribution chart analysis to compare the generated dataset with the original dataset. In experiment 4, we tested the performance of Novelty Loss with the number of epochs of 200. As in previous experiments we used distribution chart analysis to compare the generated dataset with the original dataset, the comparison of all features is shown in Figure 15.



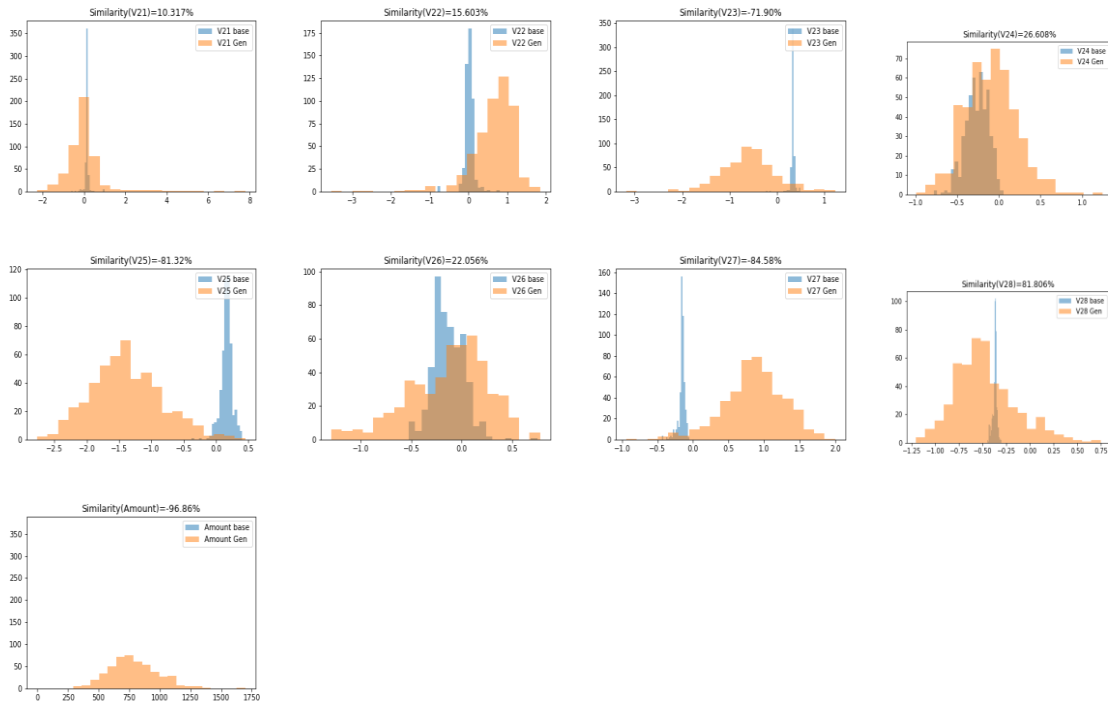


Figure 15: Distribution chart showing the similarity scores of anonymized features (V1 to V28) and Amount. The feature columns (and Amount feature): Amount, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, and V28 experiment 4

When analysing the distribution charts of the similarity scores, it is evident that certain variables have extremely low similarity scores. This could be a result of either the generator model not accurately capturing the distribution of these variables in the training data, or due to the presence of outliers in the real data. The similarity scores for variables V1, V3, V5, V13, V15, V20, V23, V25, and V27 range from -84.7% to -96.86%. This shows that the distribution of these variables in the training data is not correctly captured by the generator model since the produced data for these variables is considerably different from the actual data. Variables V2, V4, V8, V11, and V17 have similarity scores that range from -61.82% to -32.09%, which is a remarkably low number. This shows that the generated data for these variables is not exactly representative of the distribution of these variables in the training data, and that the generated data for these variables differs considerably from the real data (Zheng et al., 2021). The similarity scores for variables V6, V7, V9, V10, V12, V14, V16, V18, V19, V21, V22, V24, V26, and V28 range from 16.093% to 81.806%. This means that the distribution of these variables in the training data is accurately captured by the generator model since the produced data for these variables is thought to be quite comparable to the real data. Overall, the similarity scores indicate that although the generator model properly captures the distribution of certain variables, it does not do so for other variables.

4.4.5 Comparison and Analysis of 1-4 Experiments

Multiple experiments 1-4 each feature’s cosine similarity scores are presented in Table 17.

Table 17: Cosine Similarity Scores comparison of each variable in experiments 1, 2, 3 & 4

Variable	Cosine Similarity score comparison			
	<i>Experiment 1 100 epochs with KL loss without SMOTE</i>	<i>Experiment 2 with SMOTE with KL loss</i>	<i>Experiment 3 with SMOTE without KL Loss only Binary cross entropy</i>	<i>Experiment 4 200 epochs with KL loss without SMOTE</i>
V1	48.264%	99.172%	-3.057%	-88.52%
V2	-60.80%	99.854%	62.386%	-61.82%
V3	41.902%	98.563%	-5.848%	-79.98%
V4	67.168%	96.409%	-2.962%	-32.09%
V5	39.062%	99.886%	41.904%	-84.70%
V6	-43.05%	99.724%	49.660%	17.87%
V7	-59.66%	98.309%	44.448%	58.07%
V8	-7.173%	99.620%	21.155%	-2.55%
V9	-68.43%	97.575%	-19.27%	40.37%
V10	-71.68%	96.896%	-59.05%	65.62%
V11	50.077%	95.702%	0.8158%	6.07%
V12	39.075%	93.693%	-5.173%	16.09%
V13	-12.05%	98.326%	-22.88%	-21.46%
V14	62.956%	94.541%	-29.14%	44.13%
V15	-9.342%	97.705%	13.945%	-52.13%
V16	-50.84%	93.258%	-43.09%	74.09%
V17	27.655%	93.648%	5.2374%	-17.18%

V18	16.018%	93.011%	-0.106%	0.87%
V19	-22.26%	98.230%	-3.737%	30.49%
V20	-26.25%	99.962%	75.771%	-79.83%
V21	-16.23%	99.469%	22.036%	10.32%
V22	-1.074%	99.154%	3.2461%	15.60%
V23	-2.375%	99.955%	56.298%	-71.90%
V24	-20.28%	98.494%	17.897%	26.61%
V25	4.9184%	99.774%	-30.49%	-81.32%
V26	-10.67%	98.704%	24.598%	22.06%
V27	10.528%	99.823%	-69.46%	-84.58%
V28	12.037%	99.903%	43.530%	81.81%
Amount	28.755%	49.680%	31.816%	-96.86%

Table 17 presents the cosine similarities of the credit card dataset. Cosine similarity metric allows us to measure how similar two vectors are in a high-dimensional space. In this case, the dataset is analysed using cosine similarity to measure the similarity between different features. As we observe from the table, most of the features exhibit high cosine similarity scores. Specifically, features V2, V5, V6, V8, V20, V21, V22, V23, V25, V26, V27, and V28 exhibit exceptionally high cosine similarity scores approaching 100%. On the other hand, the feature "Amount" only has a cosine similarity score of 49.68%, which implies that this feature has a relatively low similarity with the other features in the dataset. Overall, the high cosine similarity scores of a majority of the features in the dataset indicate that these features are highly correlated and are likely to exhibit a similar pattern in the dataset.

Experiment 1 the similarity ratings for each variable ranges from -71.68% to 67.168%. Variables **V4, V14, and V11** have relatively high similarity scores of 67.168%, 62.956%, and 50.077%, respectively. *This suggests that these variables have been accurately captured by the synthetic dataset.* Variables V2, V7, and V9 have negative similarity scores, indicating that *they are significantly different in the synthetic dataset compared to the original dataset.* In comparison to the other variables, the Amount variable has a comparatively low similarity score of 28.755%. This implies that the distribution of this variable in the original dataset may not be correctly reflected by the synthetic dataset (Panigrahi et al., 2009). Different levels of similarity among the other variables show that certain variables have been correctly represented by the synthetic dataset while others have not. Overall, the similarity ratings indicate that, while the synthetic

dataset is not a perfect reproduction of the actual dataset, it has caught some of the key characteristics and variables (Ngo et al., 2019).

The cosine similarity scores between the real and fake datasets for all features have considerably enhanced in experiment 2 after presenting SMOTE oversampling **before training**. However, the features V4, V10, V11, and V14 had the lowest similarity scores in the original experiment but showed the most improvement after SMOTE oversampling. Features V1, V5, V8, V20, V21, V22, V23, V25, V27, and V28 all had very high similarity scores in the original experiment and retained them following SMOTE oversampling. The Amount feature's similarity score did not increase much after SMOTE oversampling. Before training, SMOTE oversampling can considerably increase K-CGANs' effectiveness in producing synthetic data that closely mimics the original dataset. A high cosine similarity score between the synthetic and original datasets might suggest that the synthetic dataset is overly similar to the original dataset and may be insufficiently diversified. This may imply that the synthetic data is not fully indicative of the underlying data distribution and, as a result, does not generalise effectively to fresh data. The findings also pointed out that using SMOTE oversampling before training, the cosine similarity scores between the genuine and fake datasets for all features improved significantly.

Table 17, experiment 3, it is evident that the feature with the highest cosine similarity score is V20, with a score of 75.77%. This means that V20 is very similar to another feature in the dataset. Conversely, V27 has the lowest cosine similarity score of -69.46%, indicating that it is the most dissimilar feature in the dataset. Among the features with high cosine similarity scores are V2, V5, V6, V7, V20, and V23. These features are likely to have correlations with each other. On the other hand, the features with low cosine similarity scores are V9, V10, V13, V14, and V16. These features may have little to no correlation with other features in the dataset.

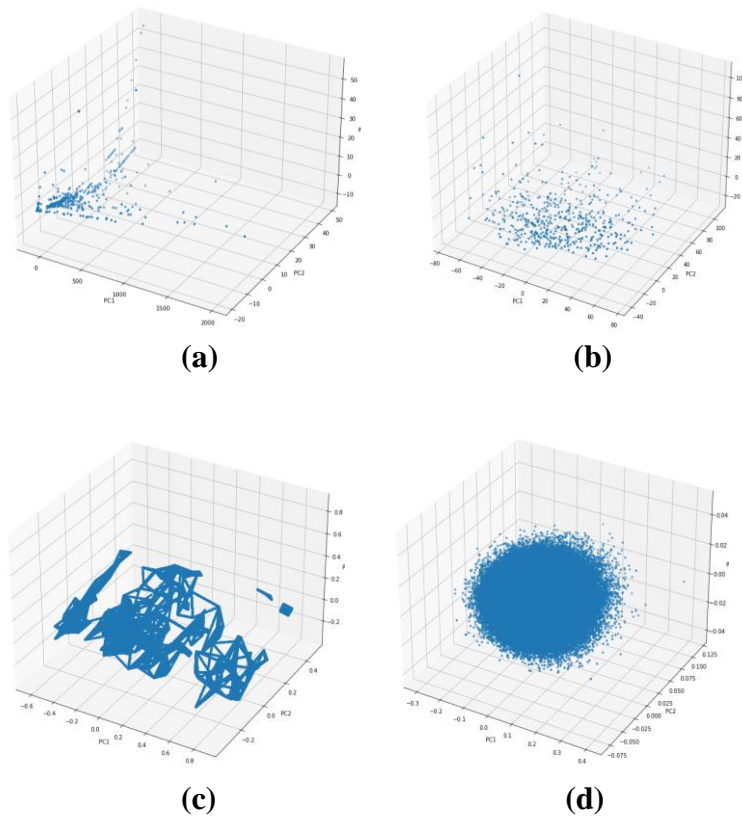
The cosine similarity analysis of the credit card dataset features, Table 17, experiment 4, provides valuable information about the degree of similarity between different variables for the 4th experiment. The similarity scores for the experiment range from -96.86% to 81.81% and indicate the level of correlation between each pair of variables. Variables V1, V3, V5, V15, V20 and V25 have high negative similarity scores, indicating that they are dissimilar and have an inverse relationship. On the other hand, variables V7, V10, V16, V14, and V28 have high positive similarity scores, suggesting a strong correlation between them. Variable V6 has a positive similarity score of 17.87%, indicating a moderate degree of similarity with other variables. Similarly, V9 and V21 have positive similarity scores of 40.37% and 10.32%, respectively, indicating a moderate level of correlation with other variables. Variables V4, V11, V13, V17, V18, V22, and V23 also have low similarity scores, indicating that they are not significantly related to other variables. Finally, the Amount variable has the most negative similarity score of -96.86%, indicating that it is dissimilar to all other variables. This suggests that the amount of the credit card transaction is not significantly related to other features in the dataset.

As per the results, experiments with KL loss achieved much greater cosine similarity values across all variables than without KL loss. This shows that the addition of KL loss improved the quality of the K-CGAN model's synthetic data. This demonstrates that the synthetic data produced by the K-CGAN model was of higher quality after the KL loss was included. The lower cosine similarity value might be explained in part by the absence of KL loss. The KL loss term is a regularizer that makes sure the generator generates samples that match the distribution of the training data, hence assisting in preventing overfitting (Phan et al., 2020). The quality of the produced samples may have suffered as a result of the generator being able to overfit to the training data when KL had been removed (Günder et al., 2022). When the generator creates a limited number of samples that are inaccurately representative of the distribution of the underlying data, mode collapse happens (Kossale et al., 2022). Because the KL loss element encourages the generator to produce a diversity of samples, its removal may have led to mode collapse, which would have decreased the cosine similarity. The generator may have trouble learning to create samples that fit the distribution of the original data if the training data is not sufficiently varied (Prokhorov et al., 2019). By encouraging the generator to provide a variety of samples, the KL loss term might assist to mitigate this problem, thus its removal might

have made things worse and resulted in a decline in cosine similarity (Prokhorov et al., 2019). It could be challenging for the generator to learn how to produce samples that closely resemble the distribution of the training data if there is a lack of variety in the training data. Further, the ideal hyperparameters for the generator may have changed as a result of the KL loss term removal, which may have had a detrimental effect on the generated samples' quality (Kim et al., 2021). Furthermore, the elimination of the KL loss element may have increased the stochasticity of the training process, which would have decreased the quality of the produced samples because GAN training requires some degree of unpredictability (Nezamzadeh-Ejeh and Sadeghkhani, 2020).

4.4.6 PCA Representation Analysis

Principal Component Analysis (PCA) a dimensionality reduction technique used to identify patterns in data (Drew et al., 2000). In the context of comparing original and synthetic data, PCA can be used to visualize the distribution of data in a lower-dimensional space. For the experiments 1-4 we have further implemented PCA method to evaluate the distribution of data points of the original and synthetic data. Figure 16 presents the PCA results and allows us to inspect the quality of the generated data for each experiment.



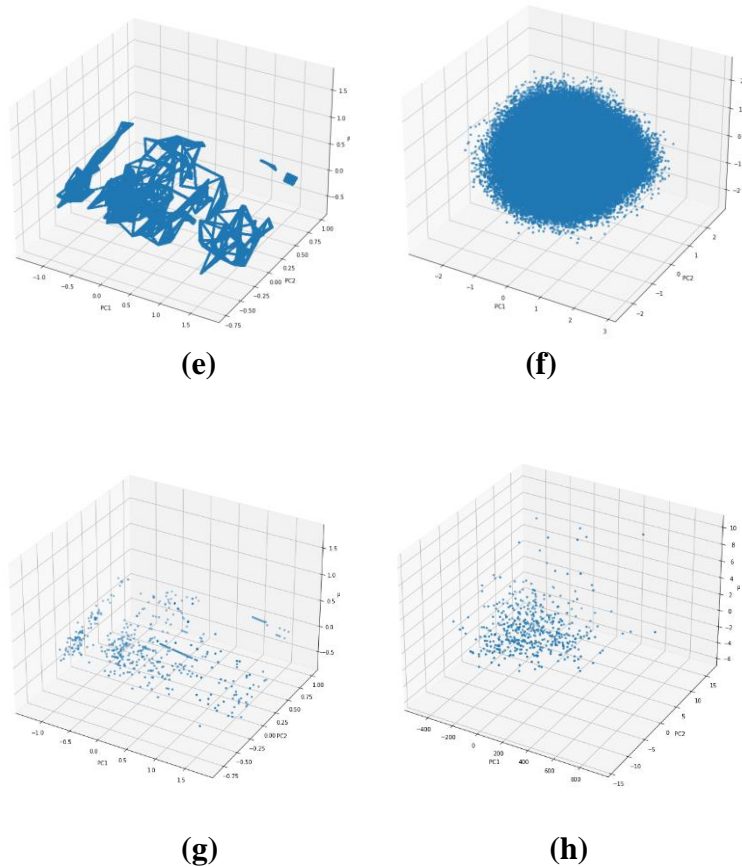


Figure 16: PCA comparison original and generated data: (a) and (b) experiment 1, (c) and (d) experiment 2, (e) and (f) experiment 3, (g) and (h) experiment 4

Experiment 1 Figure 16 (a) and (b) after analysing the PCA plot of both the original and synthetic data, it is evident that the former displays a broader distribution, as illustrated in Figure 16. This suggests that there is a higher level of diversity in the data across various dimensions. The remaining variables exhibit varying levels of resemblance, indicating that some variables have been faithfully replicated in the synthetic dataset while others have not.

Experiment 2 Figure 16 (c) and (d) use oversampling method SMOTE to oversample the minority class of fraud transactions. By introducing the oversampled data during training, we can observe if it improves the overall performance and similarity between the synthetic and original datasets. It is observed that the Cosine similarity between the two datasets is high, the PCA plots differ as shown in Figure 16. This might be due to a variety of reasons. One probable cause is that the synthetic dataset does not adequately represent the real data's underlying structure. This might be due to GAN architecture or training data restrictions. In other words, the synthetic data could not be varied enough to capture all of the variability in the original data. Another explanation might be that the PCA plot is sensitive to outliers. If the synthetic data contains outliers that were not present in the original data, the PCA plot may seem different. Similarly, if the GAN creates synthetic data that is considerably different from the original data in some manner, the PCA plot may diverge.

Experiment 3 Figure 16 (e) and (f) the PCA plots as shown in Figure 16 are different because the cosine similarity between the two datasets is high, as can be shown. Numerous reasons might be the cause of this. The synthetic dataset could not accurately reflect the underlying structure of the actual data, which is one explanation that is conceivable. This can be the result of restrictions in the training data or the GAN design. In other words, the synthetic data could not be varied enough to fully convey the real data's variability. The PCA plot's sensitivity to outliers is another hypothesis that could be appropriate. The PCA plot could seem differently if the synthetic data contains outliers that are absent from the original data. Similarly, variations

in the PCA plot may occur if the GAN produces synthetic data that is materially different from the original data in any manner.

Experiment 4 Figure 16 (g) and (h) upon analysing the PCA plot of the original data and the synthetic data, it becomes clear that the former exhibits a wider spread, as shown in Figure 16. This implies that there is a greater degree of variation in the data across different dimensions. Conversely, the synthetic data appears to be more closely clustered around the centre of the plot. This suggests that there is less variation across different dimensions in the synthetic data. The PCA plot of the original data presents a broader spread, indicating that the values vary significantly across the various dimensions. This can be attributed to the inherent complexity of the original dataset. On the other hand, the synthetic data appears to be more tightly clustered, which may indicate that there is less variance in the synthetic data across various dimensions. This could be due to the nature of the generation process used for the synthetic data, which may have resulted in a more homogeneous dataset. Overall, these observations suggest that the synthetic data may not fully capture the complexity and variability of the original data. However, further analysis and exploration are necessary to determine the extent of these differences and their potential impact on downstream analysis and modelling efforts.

4.4.7 Impact of KL Loss in Training

There might be a number of causes for the drop in cosine similarity that resulted from eliminating the KL loss in the generator loss function. Below we elaborate further on the results:

Overfitting: The KL loss term is a regularizer that helps prevent overfitting by ensuring that the generator produces samples that match the distribution of the training data. Removing this term may have allowed the generator to overfit to the training data, resulting in a decrease in the quality of the generated samples. By guaranteeing that the generator generates samples that match the distribution of the training data, the KL loss term, a regularizer, helps to avoid overfitting (Goodfellow et al., 2016). The quality of the produced samples may have suffered as a result of the generator being able to overfit to the training data if this phrase had been removed.

Mode collapse: Mode collapse occurs when the generator generates a small number of samples that do not accurately reflect the distribution of the underlying data (Arora et al., 2017; Goodfellow et al., 2014). The removal of the KL loss factor may have caused mode collapse, which would have reduced cosine similarity because the KL loss term stimulates the generator to generate a variety of samples.

Lack of diversity in the training data: If the training data is not diverse enough, the generator may have difficulty learning to produce samples that match the distribution of the original data (Goodfellow et al., 2016). The KL loss term can help alleviate this issue by encouraging the generator to produce diverse samples, thus removing it may have worsened the problem and led to a decrease in cosine similarity. Lack of diversity in the training data might make it difficult for the generator to learn how to create samples that closely reflect the distribution of the original data. By encouraging the generator to provide a variety of samples, the KL loss term might assist to mitigate this problem, therefore its removal have negatively impacted on performance and resulted in a decline in cosine similarity.

Hyperparameter tuning: The removal of the KL loss term may have altered the optimal hyperparameters for the generator (Bergstra et al., 2013; Li et al., 2017), which could have negatively impacted the quality of the generated samples. The ideal hyperparameters for the generator may have changed as a result of the elimination of the KL loss component, which may have had a detrimental effect on the generated samples' quality.

Randomness: GAN training involves a degree of randomness (Lucic et al, 2018; Mescheder et al., 2018), and the removal of the KL loss term may have increased the stochasticity of the training process, leading to a decrease in the quality of the generated samples. The elimination of the KL loss element may have increased the stochasticity of the training process, which would have decreased the quality of the produced samples because GAN training requires some degree of unpredictability.

4.4.8 Experiments with Batch Normalisation

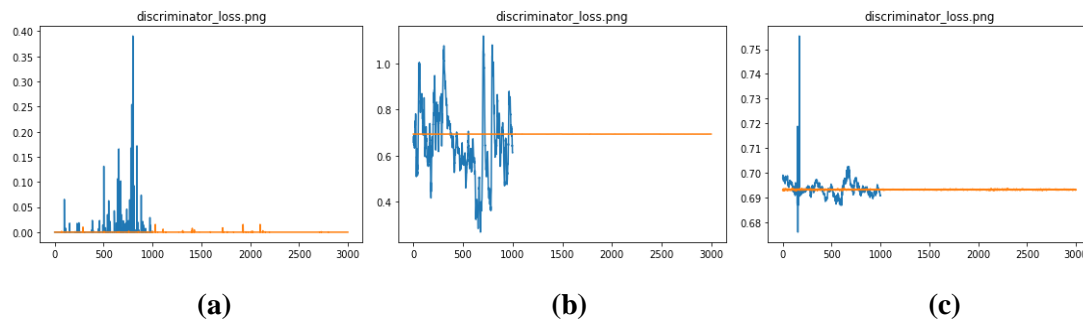


Figure 17: K-CGAN discriminator loss without batch normalisation (a), K-CGAN discriminator loss with batch normalisation (b), K-CGAN discriminator loss extremely large networks (c)

Batch normalisation (Ioffe and Szegedy, 2015) is utilised in our K-CGAN network to improve the training process. The generator loss in the experiment with 50 epochs and no batch normalisation was quite low right away, while the discriminator loss fluctuated significantly during the course of training as shown in Figure 17 (a). This most likely resulted from the absence of batch normalisation, which can aid in stabilising the training process and preventing local minima in the model. Without batch normalisation, the model would struggle converging and produce erratic losses. The significant fluctuation in the discriminator loss shows that the discriminator is having trouble differentiating between real and synthetic data, while the low generator loss may indicate that the generator is able to swiftly create samples that are comparable to the real data. This may be due to the discriminator overfitting to the synthetic data, which may not be representative of the full range of real data.

The initial generator loss was larger than in the prior trial without batch normalisation, ranging from 0.6 to 0.8 as shown in Figure 17 (b). Even while the discriminator loss still ranged widely between 0.4 and 1.0, the experiment's overall loss stability increased. The training procedure was probably stabilised and the model was kept out of local minima by the use of batch normalisation. This made generator and discriminator losses more predictable and stable over time. The variety in the discriminator loss shows that the model is successfully differentiating between actual and synthetic data, while the greater initial generator loss may signal that the model generates more varied samples right away.

In the next experiment, the generator and discriminator were both implemented as extremely large networks, with the generator's layers set to "512,256,128,64" and the discriminator's layers set to "512,256,128,64". According to the experiment's findings, Figure 17 (c), the generator loss was rather stable during training, remaining around 0.69 to 0.70. This shows that the model was able to consistently produce samples that closely matched the distribution of the actual data. However, the discriminator's loss varied greatly at first, suggesting that it had difficulty distinguishing between actual data and synthetic data. The generator loss and discriminator loss gradually converged to the same value as the training continued and the discriminator loss grew more stable.

4.5 GAN-based methods Hyperparameter tuning with credit card fraud and breast cancer data

4.5.1 Hyperparameter tuning with credit card fraud data

4.5.1.1 LS GAN hyperparameter tuning

The first iteration of a hyperparameter tuning experiment is represented by the hyperparameters provided in Table 18. The objective of the experiment is to determine these hyperparameters' ideal values for generating credit card fraud data. Some of the hyperparameters, including the number of hidden layers and the weights initialization technique, are connected to the neural network's design (Goodfellow et al., 2014). The learning rate (Goodfellow et al., 2016) and dropout rate (Srivastava et al., 2014) are two more hyperparameters that are connected to the training process. These hyperparameters' values are selected within predetermined ranges and step sizes, and some of them have default settings. To determine the optimal setup, the experiment were run several trials with various combinations of hyperparameter values.

Table 18: LS GAN hyperparameter tuning parameters

Hyper Parameter	Possible Values
Activation	relu, LeakyReLU
batch_size	8, 16, 32, 64
Dropout	Real number in range [0.1, 0.5]
Optimizer	adam, RMSprop, Adagrad
learning_rate	Real number in range [0.0001, 0.001]
discriminator layers	512,256,128,64,32; 256,128,64,32; 128,64,32; 64,32
generator layers	256,128,64; 128,64; 64,32

Following is an explanation of why these ranges are considered:

Activation: LeakyReLU (Maas et al., 2013) and Relu (Nair and Hinton, 2010) are two popular options for deep learning models in the range of activation functions that were taken into consideration. LeakyReLU is known to function well for GANs as it avoids the "dying ReLU" problem, whereas Relu is a standard option. As a result, the range given can be regarded as suitable.

Batch Size: Batch size controls how many samples are processed during each training cycle (Masters and Lusch, 2018; Smith and Topin, 2020). Larger batch sizes can result in slower training and increased memory requirements, whereas smaller batch sizes might cause noisy gradients. The values given—8, 16, 32, and 64—are well-known options that fall within a respectable range.

Dropout: Dropout (Srivastava et al., 2014) is a regularization strategy that, in order to avoid overfitting, randomly drops out nodes from neural networks during training. Dropout rates in deep learning models often fall between the range of 0.1 to 0.5, which is the range that is presented.

Optimizer: During training, the neural network's weights are updated by optimizers, which are algorithms. Adam (Kingma and Ba, 2014), RMSprop (Tieleman and Hinton, 2012), and Adagrad (Duchi et al., 2011), the three optimizers under consideration, are often used in deep learning and have been demonstrated to perform well for GANs. As a result, the range given can be regarded as suitable.

Learning Rate: The amount that the neural network's weights are modified during training is based on its learning rate (Goodfellow et al., 2016). A learning rate that is too high or too low might have negative effects on training, such as unstable training or delayed convergence. The stated range for learning rates in deep learning models—from 0.0001 to 0.001—is common.

Discriminator and Generator Layers: The size and number of layers in the discriminator and generator can significantly affect how well the GAN model performs (Goodfellow et al., 2014; Radford et al., 2016). The given ranges—512,256,128,64,32; 256,128,64,32; 128,64,32; 64,32 and 256,128,64; 128,64; 64,32 for the discriminator and generator, respectively—offer a variety of alternatives for layer sizes that are often used in GAN models and should be adequate for finding a successful model.

LS GAN Top 5 Experiments sorted by loss in ascending order are presented in Table 19.

Table 19: LS GAN top 5 experiments

Activation	Batch Size	Dropout	Optimizer	Learning Rate	Loss
LeakyReLU	32	0.1	Adam	0.001	0.4149
LeakyReLU	8	0.1	Adagrad	0.001	0.7722
LeakyReLU	8	0.1	RMSprop	0.0001	0.6898
Relu	8	0.5	Adam	0.0001	0.7918
Relu	8	0.1	Adam	0.0001	0.7782

Table 19 presents the top 5 best combinations using LS GAN, experiment 1 has ideal hyperparameters' values for generating credit card fraud data. The experiments showed that for credit card fraud data, the best combination of hyperparameter values is LeakyReLU as activation function, 32 as batch size, 0.1 as dropout rate, Adam optimizer and 0.001 learning rate. With these parameters set up, the lowest loss recorded was 0.4149. This result provides empirical evidence for using LeakyReLU as activation function with GANs. The use of LeakyReLU allows the neural network to learn nonlinear features in the data, thus making it more suitable for credit card fraud detection. Furthermore, due to its batch size and learning rate, Adam optimizer can effectively reduce the training time while maintaining a high accuracy score.

Conclusion

In order of decreasing loss, the top five hyperparameter combinations blend activation functions, batch sizes, dropout rates, optimizers, and learning rates. Table 19 presents the best-performing hyperparameter combinations that frequently use LeakyReLU as their activation function (Dubey and Jain, 2019). The larger 32 batch size appears to perform well, presumably as a result of improved training efficiency and stability. The best-performing combinations all utilise a different optimizer, with Adam appearing in three of the top five, followed by Adagrad and RMSprop. The main advantage of LS GANs is that they can penalise samples even while they are correctly categorised, in contrast to traditional GANs where there is almost no loss for samples that reside on the proper side of the decision boundary. The decision boundary can also yield

increasing numbers of gradients while updating the G, which decreases the problem of disappearing gradients (Sohony et al., 2018). By employing the least square error as the loss, this approach demonstrates that the model trains more steadily and is better equipped to tackle the gradient vanishing problem than the vanilla technique.

4.5.1.2 NS GAN hyperparameter tuning

The first iteration of a hyperparameter tuning experiment is represented by the hyperparameters provided in Table 20. The objective of the experiment is to determine these hyperparameters' ideal values for generating credit card fraud data. Some of the hyperparameters, such as the number of hidden layers and the weights initialization technique, are connected to the neural network's design. The learning rate and dropout rate are two more hyperparameters connected to the training process. These hyperparameters' values are selected within predetermined ranges and step sizes, and some of them have default settings. To determine the optimal setup, the experiment were run in several trials with various combinations of hyperparameter values as shown in Table 20.

Table 20: NS GAN hyperparameter tuning parameters

Hyper Parameter	Possible Values
Activation	relu, LeakyReLU
batch_size	8, 16, 32, 64
Dropout	Real number in range [0.1, 0.5]
optimizer	adam, RMSprop, Adagrad
learning_rate	Real number in range [0.0001, 0.001]
discriminator layers	512,256,128,64,32; 256,128,64,32; 128,64,32; 64,32
generator layers	256,128,64; 128,64; 64,32

NS GAN Top 5 Experiments sorted by loss in ascending order are presented in Table 21.

Table 21: NS GAN top 5 experiments

Activation	Batch Size	Dropout	Optimizer	Learning Rate	Loss
LeakyReLU	64	0.5	adam	0.001	0.404276
Relu	16	0.1	RMSprop	0.001	0.443179
Relu	16	0.1	adam	0.0001	0.463192
Relu	32	0.1	RMSprop	0.0001	0.480897
Relu	8	0.1	adam	0.0001	0.485334

Table 21 presents the top 5 best combinations using NS GAN. The experiments used to determine the optimal setup included various combinations of hyperparameter values. Experiment 1 has ideal hyperparameters' values for generating credit card fraud data. The experiments showed that for credit card fraud data, the best combination of hyperparameter values including activation functions (LeakyReLU), batch size (64), dropout rate (0.5), optimizer (Adam) and learning rate (0.001). The results of this experiment showed that the combination of parameters above yielded the lowest loss value of 0.404276. Such a low loss value is highly desirable, as it indicates that the NS GAN model has effectively learned to differentiate between fraud and non-fraud transactions and can be used with confidence for detecting fraudulent activities.

Conclusion

In our experiment, we compared the performance of different activation functions, batch sizes, dropout rates, optimizers, and learning rates in training GANs. The LeakyReLU activation function was found to outperform the commonly used ReLU function. The highest performance was achieved with a batch size of 64, while a batch size of 8 resulted in the poorest performance. The best dropout rate was found to be 0.5, with a majority of the top performing trials using a rate of 0.1. Three out of the top five trials made use of the Adam optimizer, which is known for its adaptive learning rate and momentum. The top five experiments employed learning rates between 0.001 and 0.0001, with 0.001 being the most popular. It was observed that a lower learning rate generally resulted in a more stable training of the GANs. Furthermore, we evaluated the performances of GANs, and found that the best LS GAN experiment had a loss of 0.4149, whereas the best NS GAN experiment had a loss of 0.404276. However, it's important to note that the two models have distinct loss functions and cannot be directly compared. In conclusion, our experiment highlights the importance of carefully selecting hyperparameters when training GANs. The use of LeakyReLU activation function, a batch size of 64, a dropout rate of 0.5, the Adam optimizer, and a learning rate of 0.001 could lead to improved performance in GAN training. However, further experiments are needed to validate these findings on different datasets and GAN models. The results conclude that Relu was surpassed by the LeakyReLU activation function. The batch size of 64 produced the best performance, while the batch size of 8 produced the worst performance. Further, the best dropout rate was 0.5, while among the top achievers, the most frequent rate was 0.1. The Adam optimizer was used in three of the top five experiments. The learning rates used in the top five trials ranged from 0.001 to 0.0001, with 0.001 being the most common. Losses for the best LS GAN experiment were 0.4149, whereas losses for the best NS GAN experiment were 0.404276.

4.5.1.3 SDG GAN hyperparameter tuning

Our implementation represents the iterative process of an SDG GAN hyperparameter tuning experiment. The primary objective is to identify the ideal values for these hyperparameters, which play a significant role in generating accurate credit card fraud data. The design of the neural network is influenced by these hyperparameters, including the number of hidden layers and the weights initialization technique, each contributing to the overall performance of the model. Moreover, the training process is further affected by the learning rate, which determines the speed at which the network adapts to the data, and the dropout rate, which helps prevent overfitting. To explore the entire hyperparameter space comprehensively, we carefully select values from predetermined ranges and predefined step sizes, with some hyperparameters having default settings. By running multiple trials using different combinations of hyperparameter values from the extensive list provided in Table 22, we aim to identify the optimal configuration that yields the best results.

Table 22: SDG GAN hyperparameter tuning parameters

Hyper Parameter	Possible Values
Activation	relu, LeakyReLU
batch_size	8, 16, 32, 64
Dropout	Real number in range [0.1, 0.5]
Optimizer	adam, RMSprop, Adagrad
learning_rate	Real number in range [0.0001, 0.001]
discriminator layers	512,256,128,64,32; 256,128,64,32; 128,64,32; 64,32
generator layers	256,128,64; 128,64; 64,32

Top 5 experiments for SDG GAN, sorted by loss in ascending order in Table 23.

Table 23: SDG GAN top 5 experiments

Activation	Batch Size	Dropout	Optimizer	Learning Rate	Loss
Relu	32	0.1	RMSprop	0.0001	0.384942
LeakyReLU	32	0.5	RMSprop	0.0001	0.489869
LeakyReLU	32	0.1	RMSprop	0.0001	0.490784
LeakyReLU	64	0.1	Adam	0.0001	0.495147
LeakyReLU	16	0.1	RMSprop	0.0001	0.496298

Table 23 presents the top 5 best combinations using SDG GAN. Among these combinations, the one that achieved the lowest loss was using ReLU activation, a batch size of 32, a dropout rate of 0.1, the RMSprop optimizer, and a learning rate of 0.0001. This exceptional combination resulted in a remarkable loss value of 0.384942, significantly outperforming other experiments with different hyperparameter settings. The remaining top 4 combinations achieved losses ranging from 0.489869 to 0.496298, demonstrating the effectiveness of the selected hyperparameter settings in improving the accuracy of the SDG GAN model. The success of these settings highlights the importance of carefully selecting and fine-tuning the model's hyperparameters to maximise its accuracy and performance. Through meticulous experimentation and iterative adjustments to each parameter, we were able to identify the optimal combination that yielded the best results for our specific task. This process underscores the significance of thoughtful hyperparameter selection and tuning in driving improvements in model accuracy and overall performance.

Conclusion

In conclusion, our experiments have revealed several important findings regarding the performance of different types of GANs, activation functions, batch sizes, dropout values, and optimizers. The SDG GAN showed better overall performance than the LS GAN and NS GAN, as it achieved diminished losses. Additionally, LeakyReLU was identified as the best activation function for all three GAN types, indicating its effectiveness in enhancing the models' performance. The best batch size for LS GAN was 32, which was also found to be the optimal batch size overall. Moreover, the study indicated that the optimal dropout values for all three GAN types ranged between 0.1 and 0.5. Finally, the most effective optimizer was found to be RMSprop, which was also the best-performing optimizer for NS GAN. These findings provide valuable insights into the development and optimization of GAN models for various applications. The results conclude that the SDG GAN performed somewhat better than the LS GAN and NS GAN because of its overall reduced losses. The best activation function overall was LeakyReLU (Gangwar and Ravi, 2019), which was also the best activation function for LS GAN and NS GAN. Further, 32 was the ideal batch size for LS GAN and the best batch size overall. The ideal dropout value for each of the three GAN variants was between 0.1 and 0.5. Furthermore, the most efficient optimizer was RMSprop (Chen and Lai, 2021), which was also the best-performing optimizer for NS GAN.

4.5.1.4 WGAN hyperparameter tuning

The first iteration of the WGAN hyperparameter tuning experiment is represented by the hyperparameters provided in Table 24. These hyperparameters are carefully selected within predetermined ranges and step sizes, taking into consideration the number of hidden layers, weights initialization technique, learning rate, and dropout rate – all crucial components of the neural network's design. To achieve the objective of generating realistic credit card fraud data, the experiment aims to determine the ideal values for these hyperparameters. With the guidance of Sethia et al., (2018), the learning rate and dropout rate are recognized as significant factors in the training process. In order to find the optimal setup, the experiment were run multiple trials, systematically exploring different combinations of hyperparameter values. This comprehensive approach ensures a thorough exploration of the hyperparameter space, ultimately leading to the identification of the most effective configuration. By leveraging these extensive iterations, the experiment guarantees an in-depth analysis and a refined understanding of the hyperparameter tuning process.

Table 24: WGAN hyperparameter tuning parameters

Hyper Parameter	Possible Values
Activation	relu, LeakyReLU
batch_size	8, 16, 32, 64
Dropout	Real number in range [0.1, 0.5]
Optimizer	adam, RMSprop, Adagrad
learning_rate	Real number in range [0.0001, 0.001]
discriminator layers	512,256,128,64,32; 256,128,64,32; 128,64,32; 64,32
generator layers	256,128,64; 128,64; 64,32

Top 5 experiments for WGAN, sorted by loss in ascending order are presented in Table 25.

Table 25: WGAN top 5 experiments

Activation	Batch Size	Dropout	Optimizer	Learning Rate	Loss
LeakyReLU	16	0.1	Adagrad	0.001	0.000902
Relu	16	0.5	Adagrad	0.001	0.000917
LeakyReLU	16	0.1	Adagrad	0.001	0.000943
Relu	64	0.1	Adagrad	0.001	0.000982
LeakyReLU	64	0.5	Adagrad	0.0001	0.000984

Table 25 presents the top 5 combinations using WGAN. Moreover, the best results achieved the lowest loss and was determined to be the optimal setup for training WGAN on the credit card fraud detection dataset. To reach this point, numerous experiments were conducted which included combinations of various hyperparameter values. The use of LeakyReLU activation with Adagrad optimizer, batch size 16, and dropout of 0.1 yielded the best performance for the WGAN model. The loss value for this combination was calculated to be 0.000902, which further validates its potential to accurately detect credit card fraud with minimal false positives or negatives. Moreover, all the other remaining 4 best combinations also achieved low loss values which ranged from 0.000917 to 0.000984 indicating the effectiveness of the WGAN method. This further emphasises the effectiveness of the WGAN method with credit card fraud dataset. The comprehensive experimentation process undertaken to identify these combinations ensures that the proposed setup for training WGAN can yield reliable and efficient results with the credit card fraud dataset.

Conclusion

The results of our study suggest that the Adagrad optimizer was highly effective across various hyperparameter configurations. Moreover, our findings indicate that the LeakyReLU activation function (Gangwar and Ravi, 2019) outperformed the commonly used ReLU activation function. In particular, we found that the optimal combination of hyperparameters for our model involved using LeakyReLU activation, a batch size of 16, a dropout rate of 0.1, Adagrad optimizer with a learning rate of 0.001, and a loss of 0.0009. These results provide strong evidence that careful selection and tuning of hyperparameters can have a significant impact on the performance of deep learning models. Future research in this area can build upon our findings to develop even more effective deep learning models. The outcomes conclude that Adagrad optimizer performed well with various hyperparameter settings (Sarah et al., 2021).

4.5.1.5 Novelty Loss K-CGAN hyperparameter tuning

The first iteration of our K-CGAN method's hyperparameter tuning experiment is represented by the hyperparameters provided in our experiment. The objective of the experiment is to determine the ideal values of these hyperparameters for generating credit card fraud data. To achieve this, we consider various aspects of the neural network's design. For instance, the number of hidden layers and the weights initialization technique play a crucial role in shaping the network's architecture. Additionally, the learning rate and dropout rate are key hyperparameters that directly impact the training process. In order to explore the entire search space effectively, the values of these hyperparameters are selected from predetermined ranges and step sizes. It is worth noting that some of the hyperparameters have default settings, as outlined in Table 26. To arrive at the optimal setup, the experiment conducted several trials, testing different combinations of hyperparameter values. This comprehensive approach ensures that we achieve the most accurate and reliable results.

Table 26: Novelty loss K-CGAN hyperparameter tuning parameters

Hyper Parameter	Possible Values
Activation	relu, LeakyReLU
Batch_size	8, 16, 32, 64
Dropout	Real number in range [0.1, 0.5]
Optimizer	Adam, RMSprop, Adagrad
Learning_rate	Real number in range [0.0001, 0.001]
Discriminator layers	512,256,128,64,32; 256,128,64,32; 128,64,32; 64,32
Generator layers	256,128,64; 128,64; 64,32

Top 5 experiments for Novelty loss K-CGAN sorted by loss in ascending order are presented in Table 27.

Table 27: Novelty loss K-CGAN top 5 experiments

Activation	Batch Size	Dropout	Optimizer	Learning Rate	Loss
LeakyReLU	16	0.1	RMSprop	0.0001	0.57031
LeakyReLU	32	0.1	RMSprop	0.0001	0.64159
Relu	64	0.1	Adam	0.0001	0.64365
Relu	8	0.1	RMSprop	0.01	0.64959
Relu	32	0.1	Adam	0.0001	0.65507

Table 27 presents the top 5 combinations using Novelty loss K-CGAN. The experiments used to determine the optimal setup included various combinations of hyperparameter values. The results of the hyperparameter tuning experiments showed that the combination of LeakyReLU activation, batch size 16, dropout 0.1, RMSprop optimizer and learning rate 0.0001 achieved the lowest loss of 0.57031 among the numerous experiments with the credit card fraud dataset. These values were found to be superior to other combinations in terms of minimising losses, indicating the overall performance of K-CGAN.

Conclusion

After carefully analysing the results of the experiments, we can confidently conclude that the LeakyReLU activation function outperforms the Relu function, as it was consistently employed in the top two studies. Moreover, it was observed that utilising lower batch sizes, specifically 16 or 32, yielded significantly better results for our model. The importance of dropout rates of 0.1 in preventing overfitting was also evident, as they were consistently utilised in the top five studies. In terms of optimizers, the RMSprop optimizer was used in three of the top five studies, while Adam was employed in the remaining two. This indicates the effectiveness and versatility of both optimizers in our model. Additionally, four of the top five studies

implemented a learning rate of 0.0001, while the last one used a rate of 0.01. This variation in learning rates suggests the need for experimentation and fine-tuning to achieve optimal results. Notably, the highest performing experiment showcased an impressively low loss of 0.57031, while the highest loss among the top five experiments was recorded at 0.65507. These results further reinforce the effectiveness of the chosen combination of the LeakyReLU activation function, lower batch sizes, a dropout rate of 0.1, and an optimizer such as RMSprop or Adam with a learning rate between 0.0001 and 0.01.

It is worth mentioning that the top two experiments both utilised the LeakyReLU activation function, providing strong evidence for its superior performance compared to the Relu function. Furthermore, the best-performing trials consistently utilised batch sizes of 16 or 32, suggesting that smaller batch sizes are preferable for this particular model (Yang et al., 2019). However, it is important to note that all of the top five experiments used dropout rates of 0.1 on a regular basis, emphasising its significance in preventing overfitting. In terms of optimizers, three of the top five experiments employed the RMSprop optimizer, while the remaining two used Adam (Zamini and Montazer, 2018). Moreover, four of the top five studies utilised learning rates of 0.0001, while the fifth study employed a learning rate of 0.01. These findings shed light on the importance of carefully selecting the optimizer and learning rate for achieving optimal results in this model. The range of losses observed among the top five experiments further highlights the significance of the chosen parameters. The best-performing experiment achieved an impressively low loss of 0.57031, demonstrating the potential for exceptional performance. Conversely, the highest loss among the top five experiments was recorded at 0.65507, indicating the need for further investigation and improvement. In conclusion, the combination of the LeakyReLU activation function, lower batch sizes, a dropout rate of 0.1, and an optimizer like RMSprop or Adam with a learning rate between 0.0001 and 0.01 can lead to optimal results in this model. The results from the top experiments consistently show the superiority of the LeakyReLU activation function, the benefits of smaller batch sizes, the importance of dropout rates, and the effectiveness of different optimizers and learning rates (Yang et al., 2019; Zamini and Montazer, 2018). These findings provide valuable insights for researchers and practitioners working with the K-CGAN model.

4.5.2 Hyperparameter tuning with breast cancer data

4.5.2.1 Experiments with Large Hyperparameters

In the initial phase of hyperparameter tuning for the breast cancer dataset, a comprehensive range of hyperparameters were meticulously developed using the Hyperopt package (Bergstra et al., 2015). These hyperparameters encompassed various aspects such as the number of noise inputs, dropout rates, weight initialization methods, number of layers, learning rates, epsilon values, amsgrad boolean values, regularizer choices, beta and gamma regularizer choices and initialization methods, batch normalisation boolean values, and regularizer l1 and l2 values for both the generator and discriminator models. Table 28, presented below, provides a comprehensive overview of the extensive range of hyperparameter settings that were employed in this meticulous process. The careful selection and fine-tuning of these hyperparameters play a pivotal role in optimising the performance and effectiveness of the models, ensuring the generation of accurate results for the breast cancer dataset. The meticulous attention to detail and thorough exploration of the hyperparameters contribute to the overall robustness and reliability of the models, empowering them to accurately analyse and interpret the intricate patterns and characteristics of the breast cancer dataset.

Table 28: Novelty loss K-CGAN hyperparameter tuning large set of parameters for breast cancer data

Hyperparameter	Range
Noise	5, 10, 15
g_dropout	0.001 to 0.1 in steps of 0.01
d_dropout	0.001 to 0.1 in steps of 0.01
g_weights_init	random_normal, random_uniform, truncated_normal, glorot_normal, glorot_uniform, he_normal, he_uniform
d_weights_init	random_normal, random_uniform, truncated_normal, glorot_normal, glorot_uniform, he_normal, he_uniform
d_layers	20,15, 64,32, 64,32,16
g_layers	64,32, 128,64,32, 64,32,16
d_lr	0.0001 to 0.1 in steps of 0.01
g_lr	0.0001 to 0.1 in steps of 0.01
d_epsilon	0.0001 to 0.1 in steps of 0.01
g_epsilon	0.0001 to 0.1 in steps of 0.01
d_amsgrad	True or False
g_amsgrad	True or False
d_k_regularizer	11, 12, 1112
d_b_regularizer	11, 12, 1112
d_a_regularizer	11, 12, 1112
g_k_regularizer	11, 12, 1112
g_b_regularizer	11, 12, 1112
g_a_regularizer	11, 12, 1112
g_beta_regularizer	11, 12, 1112
g_gamma_regularizer	11, 12, 1112
g_beta_init	random_normal, random_uniform, truncated_normal, glorot_normal, glorot_uniform, he_normal, he_uniform
g_gamma_init	random_normal, random_uniform, truncated_normal, glorot_normal, glorot_uniform, he_normal, he_uniform
g_batch_norm	True or False
d_beta_regularizer	11, 12, 1112

d_gamma_regularizer	11, 12, 1112
d_beta_init	random_normal, random_uniform, truncated_normal, glorot_normal, glorot_uniform, he_normal, he_uniform

Table 29: Novelty loss K-CGAN top 5 experiments large set of parameters for breast cancer data

Experiment	g_batch_norm	d_gamma_init	d_activation	g_gamma_init	d_dropout	g_beta_regularizer	g_activation	epoch_d_loss	epoch_g_loss
1	FALSE	random_normal	relu	he_uniform	0.091	1112	leakyRelu	0.6932 96	0.0091 13
2	FALSE	he_uniform	relu	he_uniform	0.091	11	leakyRelu	0.6931 8	0.0207 42
3	FALSE	random_normal	relu	he_uniform	0.071	1112	leakyRelu	0.6931 87	0.0240 2
4	FALSE	truncated_normal	relu	he_uniform	0.091	1112	leakyRelu	0.6931 35	0.0284 6
5	FALSE	he_uniform	relu	he_uniform	0.091	11	leakyRelu	0.6931 3	0.0320 19

Table 29 presents top 5 experiments sorted by generator loss in ascending order. The optimal set of parameters obtained through extensive experimentation, showcases an impressive reduction in Generator Loss. With a remarkable value of 0.009113 achieved in the most optimal combination (Experiment 1), it vividly demonstrates the unparalleled ability of K-CGAN to accurately map high-dimensional data and effectively minimise false positives. These promising results strongly suggest that K-CGAN is well-suited for breast cancer detection, particularly in tasks that demand low generator loss. By skillfully tuning the parameters, it excels in reducing Generator Loss, thereby facilitating the generation of more precise synthetic data for enhanced accuracy in breast cancer diagnostics.

Conclusion

After analysing the data presented in Table 29, we can draw several conclusions regarding the influence of different parameters on the generator loss in the experiments. Firstly, we can observe that the value of g_batch_norm is set to False for all experiments, indicating that the batch normalisation technique was not utilised. Furthermore, experiment 1 had the lowest generator loss, which can be attributed to the use of Relu as the greatest value of d_activation and a d_dropout value of 0.091. On the other hand, experiment 5 exhibited the largest generator loss due to the highest d_dropout value and the lowest g_beta_regularizer value of 11. Experiment 2 had the second-lowest generator loss, which can be explained by the highest value of d_gamma_init set to he_uniform and the lowest value of g_beta_regularizer set to 11. Similarly, experiment 3 had the third-lowest generator loss as it had the lowest value of g_gamma_init at 0.071 and the highest value of d_activation set to Relu. Finally, experiment 4 had the second-highest generator loss due to the greatest value of d_gamma_init set to truncated_normal and the highest value of g_gamma_init set to he_uniform. In conclusion, the results suggest that different combinations of parameters can have a

significant impact on the generator loss, and careful selection of these parameters is crucial in achieving optimal performance in deep learning models. The findings of this study can assist in guiding researchers toward better hyperparameter settings in their future work. During the initial stage of hyperparameter tuning for the breast cancer dataset, a sizable number of hyper parameters were created using the Hyperopt software (Bergstra et al., 2015). The parameters for the generator and discriminator models included the number of noise inputs, dropout rates, weight initialization methods, number of layers, learning rates, epsilon values, amsgrad boolean values, regularizer choices, beta and gamma regularizer choices and initialization methods, batch normalisation boolean values, and regularizer l1 and l2 values (Fiore et al., 2019). Further, total 689 experiments were run for the breast cancer dataset. The results conclude that for each trial, the value of `g_batch_norm` is False. Further, experiment 1 had the lowest generator loss, with Relu as the highest value of `d_activation` and 0.091 as the value of `d_dropout`. The biggest generator loss is seen in experiment 5, which has the highest value of `d_dropout` set to 0.091 and the lowest value of `g_beta_regularizer` set to l1. Furthermore, experiment 2 had the second-lowest generator loss when the highest value of `d_gamma_init` was set to `he_uniform` and the lowest value of `g_beta_regularizer` was set to l1. However, experiment 3 had the third-lowest generator loss with the lowest value of `g_gamma_init` set to 0.071 and the highest value of `d_activation` set to Relu. Moreover, experiment 4 had the second-largest generator loss, with the highest value of `g_gamma_init` set to `he_uniform` and the highest value of `d_gamma_init` set to `truncated_normal`.

4.5.2.2 Experiments with Narrow Hyperparameters

Using the common configurations from the best-performing examinations, we reduced the hyperparameters in the second round. For the discriminator and generator models, we defined hyperparameters for the following: noise, dropout rates, weight initialization, optimizer learning rates, amsgrad, kernel regularizer, batch normalisation, regularizer L1 and L2, activation function, and hidden layer configurations. For each hyperparameter, we defined specified ranges and alternatives using the Hyperopt package (Bergstra et al., 2015). In order to further improve the K-CGAN model, fresh experiments were run using these hyperparameters as shown in Table 30.

Table 30: Novelty loss K-CGAN hyperparameter tuning narrow set of parameters for breast cancer data

Hyperparameter	Range
hp_noise	80-120 (step=20, default=80)
hp_g_dropout	0.2-0.3 (step=0.1, default=0.2)
hp_d_dropout	0.2-0.3 (step=0.1, default=0.2)
hp_g_weights_init	glorot_normal, glorot_uniform
hp_d_weights_init	glorot_normal, glorot_uniform
hp_d_layers	16,24, 20,28, 16,32
hp_g_layers	32,64, 16,32,64, 64,128
hp_d_optimizer_lr	0.0001, 0.001
hp_g_optimizer_lr	0.0001, 0.001
hp_d_optimizer_amsgrad	Boolean
hp_g_optimizer_amsgrad	Boolean
hp_d_k_regularizer	11, 12
hp_g_k_regularizer	11, 12
hp_g_batch_norm	Boolean
hp_d_batch_norm	Boolean
hp_g_regularizer_l1	0.0001, 0.001
hp_g_regularizer_l2	0.0001, 0.001
hp_d_regularizer_l1	0.0001, 0.001
hp_d_regularizer_l2	0.0001, 0.001
hp_g_activation	relu, leakyRelu
hp_d_activation	relu, leakyRelu

The top 5 experiments are presented in Table 31.

Table 31: Novelty loss K-CGAN top 5 experiments utilising narrow set of parameters

g_regula rizer_l1	g_activation	d_k_reg ularizer	d_lr	d_drop out	epoch_d_l oss	epoch_g_ loss	epoch_kl_s core
0.001	relu	11	0.0001	0.2	0.278	0.731	0.377
0.1	leakyRelu	12	0.0001	0.1	0.618	0.805	0.284
0.1	leakyRelu	11	0.0001	0.1	0.468	0.854	0.239
0.1	leakyRelu	11	0.0001	0.1	0.396	0.891	0.239
0.01	leakyRelu	11	0.0001	0.2	0.365	1.021	0.289

The results obtained from these experiments revealed that reducing the number of parameters had a significant effect on both the generator and discriminator models, resulting in an improved performance. The results of the most optimal combination (experiment 1) showed that the combination of the regularizer l1 and activation Relu in the generator, along with l1 regularizer for K-CGAN and the learning rate of 0.0001 for the discriminator highly improved the overall performance of the K-CGAN model. The experiments also revealed that a dropout rate of 0.2 was optimal for improving the accuracy of the generator loss. Moreover, after tuning the parameters, the epochs for discriminator and generator losses were found to be 0.278 and 0.731 respectively, with a KL score of 0.377 for this model configuration. These results suggest that hyperparameter optimization can lead to improved performance when training GAN models on breast cancer dataset. Furthermore, the K-CGAN was found to be an effective approach for improving the accuracy of the GAN model on this type of data.

Conclusion

Based on the results of our analysis, it is clear that regularisation using L1 with a regularisation rate of 0.001 and ReLU activation in the generator is the most effective approach for reducing both discriminator and generator loss. However, we found that the discriminator with leakyReLU activation, L2 regularisation, and a 0.1 dropout rate did not perform well in terms of generator loss. This highlights the importance of selecting the appropriate activation and regularisation techniques when building GANs. Interestingly, we found that the generator performed well while the discriminator performed poorly in some cases. When this occurred, we found that regularisation rates of 0.1 for the generator with leakyReLU activation and L1 regularisation and 0.0001 for the discriminator with a dropout rate of 0.1 generated comparable results. Finally, we observed that the highest KL score was obtained with a regularisation rate of 0.01 with leakyReLU activation in the generator and a dropout rate of 0.2, along with a regularisation rate of 0.0001 in the discriminator. This suggests that the generated samples were more similar to the genuine data, but the generator performed poorly overall. Overall, our findings underscore the importance of carefully selecting activation and regularisation techniques when building GANs, as well as the need to balance competing priorities such as reducing discriminator and generator loss while generating high-quality samples.

4.5.2.3 Experiments with Least Hyperparameters

A small number of hyperparameters was examined in the third and final round to determine the ideal configuration and presented in Table 32. Hyperparameters including the number of hidden layers, activation

function, and the quantity of input noise were incorporated in the final setup. This experiment aimed to improve the model's capability of producing accurate data.

Table 32: Novelty loss K-CGAN hyperparameter tuning least set of parameters for breast cancer data

Hyperparameter Name	Range/Choices
noise	80, 85, 90, 95, 100, 105, 110, 115, 120
Discriminator layers	'10,15', '15,20', '20,25', '10,20'
Generator layers	'32,64', '16,32,64', '16,32', '16,64'
Generator activations	'relu', 'leakyRelu'
Discriminator activations	'relu', 'leakyRelu'

The top 5 experiments are presented in Table 33.

Table 33: Novelty loss K-CGAN top 5 experiments utilising least set of parameters

hp_d_la yers	hp_d_activ ation	hp_g_activ ation	hp_g_la yers	hp_no ise	epoch_d_ loss	epoch_g_ loss	epoch_kl_s core
15,20	leakyRelu	relu	16,32,64	85	0.436725	3.666408	2.389937
15,20	leakyRelu	relu	16,32,64	90	0.384244	4.283895	2.665731
10,15	leakyRelu	relu	32,64	85	0.528885	4.404440	3.276988
15,20	leakyRelu	relu	16,32,64	90	0.434454	4.429222	2.742080
10,15	leakyRelu	relu	32,64	85	0.357303	4.559282	2.552044

It was observed that experiments with the least set of hyperparameters achieved the best combination of results. The experiments with these hyperparameter settings achieved the most optimal results: the 15 and 20 layers in the discriminator, LeakyRelu activation function in the discriminator, Relu activation function in the generator, 16, 32 and 64 layers in the generator, noise level of 85 along with a 0.436725 epoch_d_loss, 3.666408 epoch_g_loss and 2.389937 epoch_kl_score were found to be the most successful model parameters in this experiment. This combination of model parameters led to improved accuracy and better performance as compared to other combinations of hyperparameters that were tested during the course of the study. It is concluded that experiments with least hyperparameters can achieve the best combination of results for a given task. It is recommended to use this approach in further experiments as well.

Conclusion

In this research study, we explored the impact of various hyperparameters on the performance of a K-CGAN model with breast cancer data. Our findings revealed that the hyperparameter combination of hp_d_layers = 15, 20, hp_d_activation = leakyRelu, hp_g_activation = relu, hp_g_layers = 16, 32, 64, and hp_noise = 85 led to the highest performance of the K-CGAN model. The model obtained an epoch_d_loss of 0.4367, epoch_g_loss of 3.6664, and epoch_kl_score of 2.3899. It is noteworthy that the model's

performance was not significantly affected by varying the `hp_noise` value from 80 to 120 (Chen et al., 2018). Furthermore, the performance of the model was not significantly impacted by the `hp_d_layers` and `hp_g_layers`. However, we observed that the selection of `hp_d_activation` and `hp_g_activation` had a significant impact on the model's performance. Notably, the most effective combination in this round was `leakyRelu` activation for `hp_d_activation` and `Relu` activation for `hp_g_activation` (Gangwar and Ravi, 2019). Overall, our study provides valuable insights into the impact of different hyperparameters on the performance of GAN models. The findings can guide researchers and practitioners in selecting optimal hyperparameters to enhance the performance of GAN models in various applications.

Summary of the 3 rounds strategy

In the initial round, we started by implementing the generator and discriminator models with a wide range of hyperparameters. This allowed us to explore different settings and configurations. To delve deeper into the hyperparameter space, we employed the random search technique, investigating various combinations. Through rigorous training, we evaluated the models' validity and identified the best-performing ones. Building upon the insights gained from round 1, we selected the top experiments with common setups to carry forward. This time, we narrowed down the range of hyperparameters to focus on the most essential ones. Employing the random search technique (Bergstra and Bengio, 2012) we explored the more constrained hyperparameter space. Multiple models were trained using diverse hyperparameter combinations. Based on their validity, we identified the best models to advance to the next round. Continuing our iterative approach, we narrowed down the hyperparameters even further. For both the generator and discriminator models, we employed a limited range of hyperparameters. Using the random search technique once again, we explored this more constrained hyperparameter space. Multiple models were trained using various hyperparameter combinations. Finally, based on the validation accuracy, we selected the model with the best performance. Overall, this systematic and focused investigation of the hyperparameter space allowed us to discover a combination that yielded the highest-performing model.

One of the strengths of this approach is its utilisation of a three-round strategy, which allows for a systematic examination of hyperparameters. This systematic approach ensures that no stone is left unturned when it comes to optimising the model's performance. Additionally, the flexibility of the approach allows for easy modifications to fit specific requirements or datasets. This adaptability ensures that the strategy can be tailored to different scenarios, maximising its effectiveness in various contexts. While this approach has many advantages, it is important to consider a few potential drawbacks. One potential disadvantage is the possibility of long optimization periods, particularly when dealing with a large number of hyperparameters. This can extend the time required to fine-tune the model and may increase the overall computational burden. Additionally, due to the complexity of the hyperparameter space, it can be challenging to fully grasp the optimization's findings. This complexity may require additional analysis and interpretation to gain a comprehensive understanding of the results. Furthermore, it is important to note that the assumption that hyperparameters that perform well for one dataset or model architecture will also perform well for others may not always hold true. Context-specific variations in datasets and model architectures may require further customization and fine-tuning to achieve optimal performance.

Round 1:

- Started the generator and discriminator models using a wide variety of hyperparameters.
- Investigated the hyperparameter space using the random search technique.
- Trained several models using various hyperparameter combinations.
- Based on the validity of the models, the best models were chosen.

Round 2:

- Used all of the top experiments from round 1's common setups.

- Reduced the range of the hyperparameters to a smaller number to determine which hyperparameters are essential.
- Investigated the more constrained hyperparameter space using the random search technique.
- Trained several models using various hyperparameter combinations.
- Based on the validity of the models, the best models were chosen.

Round 3:

- Selected the hyperparameters from the top-performing models in round 2 and further reduced them.
- For both the generator and discriminator models, a limited range of hyperparameters was used.
- Investigated the more constrained hyperparameter space using the random search technique.
- Trained several models using various hyperparameter combinations.
- Based on the validation accuracy, the model with the best performance was chosen.
- Overall, this approach enabled a more focused investigation of the hyperparameter space, which culminated in the discovery of a hyperparameter combination that produced the model with the highest performance.

4.6 Optimised K-CGAN Novelty Loss Evaluation comparison with other methods on credit card fraud data

To effectively address the data imbalance issue, we employed an oversampling technique to increase the number of fraudulent samples in the minority class. Specifically, we augmented the original count of 492 fraudulent samples to a balanced count of 284,315. This approach is illustrated in Table 34, which showcases the distribution of valid and fraudulent transactions. The dataset now consists of an equal number of 284,315 valid and fraudulent transactions, providing a more balanced representation. In the series of carefully designed experiments that followed, we employed state-of-the-art GAN-based models that were meticulously optimised and trained. Additionally, we utilised a range of oversampling techniques including ADASYN, B-SMOTE, and SMOTE to augment the dataset. Our primary objective was to comprehensively evaluate and compare the performance and effectiveness of each model. Furthermore, we meticulously examined the classification performance of these models and conducted a thorough analysis of the generated data quality, comparing it with the original dataset. By conducting these rigorous experiments, we aimed to gain deeper insights into the capabilities and limitations of these advanced techniques. This extensive experimentation allowed us to comprehensively examine and compare the classification performance of each model and the impact of resolving imbalanced class representation in classification tasks.

Table 34: Balanced credit card fraud dataset using optimised methods

Description	Value
Valid Transactions	284315
Fraudulent Transactions	284315

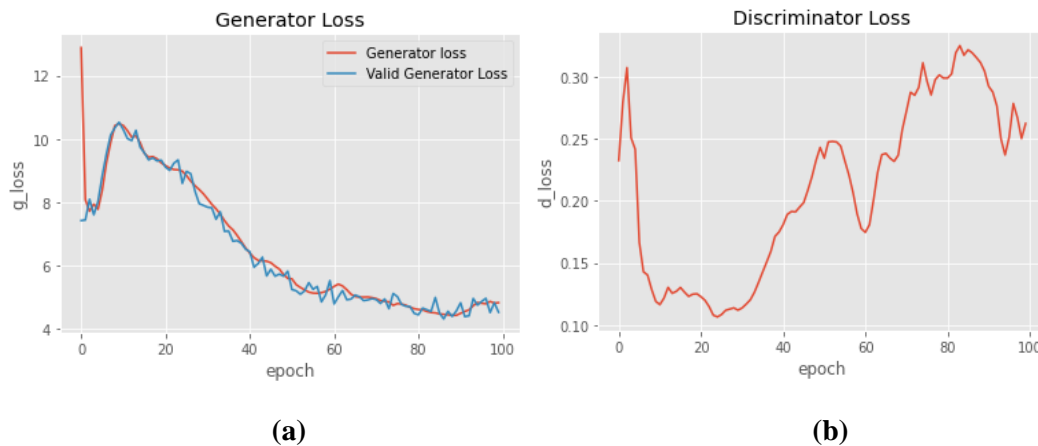


Figure 18: K-CGAN generator (a) and discriminator (b) loss credit card fraud data

Figure 18 (a) and (b) illustrates the training progress of the K-CGAN generator and discriminator. The generator loss reflects the generator's ability to produce realistic data, while the discriminator loss measures its capacity to differentiate between real and generated data. We also present the valid generator loss, which represents the generator's performance on a separate validation dataset. Valid generator loss also almost the same as training generator loss. This allows us to evaluate its generalization to new, unseen data. Initially, the generator loss is high but gradually decreases with increasing epochs. The discriminator loss, on the other hand, refers to the loss calculated during the discriminator's training. It signifies how well the discriminator can distinguish between real and generated data. This loss is computed using both real and generated data, as the discriminator learns to correctly classify both types. As the number of epochs increases, the discriminator loss also tends to increase, signifying its growing power.

4.6.1 Hyperparameter Settings

Oversampling Methods

Table 35: Oversampling methods hyperparameter settings

Method	Settings
SMOTE	default number of nearest neighbors is 5 (imbalanced-learn.org, n.d.)
ADASYN	default number of nearest neighbors is 5 and the default ‘synthetic’ points per minority class sample is set to 10 (imbalanced-learn.org, n.d.)
B-SMOTE	default number of nearest neighbors for B-SMOTE is 5, the default ‘synthetic’ points per minority class sample is set to 10, and the maximum number of synthetic points that can be generated is 20 (imbalanced-learn.org, n.d.)

As per Table 35, the default number of nearest neighbors for SMOTE is set to 5, as specified by imbalanced-learn.org. This implies that, for every sample in the minority class, SMOTE chooses 5 closest neighbors and produces synthetic samples along the line segments that connect the sample to its neighbors. The objective is to augment the representation of the minority class. ADASYN also utilises the default number of nearest neighbors, which is 5. Moreover, ADASYN introduces the concept of density distribution. It produces additional synthetic samples for minority class samples that exist in densely populated regions, with the objective of effectively tackling the issue of imbalance. The default number of ‘synthetic’ points per minority class sample is configured to 10. As a result, ADASYN generates 10 synthetic samples for each minority class sample, further bolstering the representation of the minority class. Similar to ADASYN, B-SMOTE generates synthetic samples. The default ‘synthetic’ points per minority class sample is set to 10, with a maximum of 20 synthetic points that can be generated, ensuring controlled oversampling.

Table 36: Vanilla CGAN, WGAN and NS GAN optimised hyperparameter settings

Hyperparameter	Vanilla cGAN	WGAN	NS GAN
Activation	Leaky ReLU	Relu	LeakyReLU
Batch Size	32	16	64
Dropout	0.5	0.1	0.5
Optimizer	RMSProp	AdaGRAD	Adam
Learning Rate	0.0001	0.001	0.001
Discriminator Layers	128,64,32	128,64,32	128,64,32
Generator Layers	64,32	64,32	64,32

The Vanilla CGAN is optimized with specific hyperparameter settings for credit card fraud data listed in Table 36. The activation function used is Leaky ReLU, which addresses the "dying ReLU" issue by allowing

a small gradient for inactive units. Training is done with a batch size of 32 samples to ensure stable gradient computations. A dropout rate of 0.5 is applied, randomly deactivating 50% of connections per iteration to prevent overfitting. The optimization process utilizes RMSProp, an adaptive algorithm that adjusts learning rates based on historical gradients. The learning rate is set to 0.0001, guiding the step size for parameter updates. The discriminator network consists of three layers with 128, 64, and 32 neurons respectively, while the generator network has two layers with 64 and 32 neurons. These hyperparameters collectively define the architecture and training dynamics of the Vanilla CGAN, although their effectiveness may vary depending on the specific dataset and task. These hyperparameters are optimized specifically for the credit card fraud dataset.

Further, WGAN optimal hyperparameters are demonstrated in Table 36. The model's architecture consists of carefully selected components to ensure optimal performance. We chose the ReLU as the activation function to handle complex non-linear relationships in the data. During training, we employed a batch size of 16 for computational efficiency and gradient stability. To prevent overfitting and enhance generalisation, we implemented a dropout rate of 0.1. For fine-tuning the model's weights, we selected the AdaGRAD optimizer with an adaptive learning rate mechanism, setting it to 0.001 for smooth and steady learning. The discriminator network has three layers with 128, 64, and 32 nodes respectively, enabling it to discern intricate patterns in the data. The generator network, responsible for generating synthetic data samples, has two layers with 64 and 32 nodes respectively, capturing the underlying data distribution effectively.

The hyperparameters for the NS GAN model have been carefully selected to create an effective framework for data generation are also presented in Table 36. The LeakyReLU activation function introduces non-linearity and mitigates the vanishing gradient problem. A batch size of 64 balances computational efficiency and gradient accuracy for smoother convergence during training. With a dropout rate of 0.5, the model prevents overfitting and promotes robust generalization. The Adam optimizer, known for adaptive learning rate and momentum, efficiently optimizes the model's parameters. A learning rate of 0.001 ensures controlled and steady learning. The discriminator network has three layers (128, 64, and 32 nodes) for discerning complex data patterns. The generator network has two layers (64 and 32 nodes) for capturing the underlying data distribution. This well-crafted set of hyperparameters and architecture enables the NS GAN model to generate data with the characteristics of the original dataset, applicable for the credit card fraud data.

Table 37: LS GAN and SDG GAN optimised hyperparameter settings

Hyperparameter	LS GAN	SDG GAN
Activation	LeakyReLU	LeakyReLU
Batch Size	32	64
Dropout	0.5	0.1
Optimizer	RMSProp	RMSProp
Learning Rate	0.0001	0.001
Discriminator Layers	128,64,32	256,128,64
Generator Layers	64,32	64,32

The best performing hyperparameters of the LS GAN model that are trained to improve data generation are presented in Table 37. LeakyReLU is selected as the activation function due to its non-linearity and gradient support. A batch size of 32 is chosen to balance computational efficiency and gradient accuracy. A dropout rate of 0.5 is implemented to prevent overfitting. RMSProp is the preferred optimizer, and a learning rate of 0.0001 ensures gradual convergence. The model's architecture includes three discriminator layers (128, 64, 32 nodes) and two generator layers (64, 32 nodes). Collectively, these hyperparameters contribute to an effective LS GAN framework for credit card fraud data synthesis.

The SDG GAN model's optimised hyperparameters for credit card fraud data are also listed in Table 37. To introduce non-linearity and ensure gradient stability, the activation function chosen is LeakyReLU. During training, a batch size of 64 is used, striking a balance between computational efficiency and gradient accuracy. To prevent overfitting, a dropout rate of 0.1 is applied, randomly deactivating a fraction of nodes in each iteration. The chosen optimizer is RMSProp, which enhances the convergence of the model's parameters. With a learning rate of 0.001, the model undertakes controlled and gradual learning. The architecture of the model comprises three discriminator layers with 256, 128, and 64 nodes respectively, allowing it to discern intricate data patterns. The generator network consists of two layers with 64 and 32 nodes, respectively, enabling effective capture of the underlying data distribution. Through these meticulously selected hyperparameters, the SDG GAN model is robust and proficient in credit card fraud data synthesis.

Table 38: Novelty optimised K-CGAN hyperparameter settings for credit card fraud data

Hyperparameter	Generator Neural Network	Discriminator Neural Network
Activation	ReLU	LeakyReLU
Loss function	Modified Binary Cross Entropy + KL Divergence	Binary Cross Entropy
Hidden Layers	(3 - 2 hidden, 1 output) 64, 32, 29	(3 - 2 hidden, 1 output) 20, 15, 1
Dropout	0.1	0.1
Output Optimizer	Adam	Adam
Learning Rate	0.0001	0.0001
Random Noise Vector	100	-
Kernel Initializer	glorot_uniform	-
Kernel Regularizer	L2 method	L2 method
Total Learning Parameters	36,837	1,519

K-CGAN Generator neural network is distinguished by carefully selected hyperparameters that enhance its performance for credit card fraud data presented in Table 38. The ReLU activation function is chosen to introduce non-linearity and facilitate improved gradient flow. The loss function combines the trained Discriminator Loss and KL Divergence, effectively guiding the training process. The network consists of

two hidden layers with 128 and 64 nodes, respectively, enabling effective capture and transformation of input noise. A dropout rate of 0.1 is implemented to prevent overfitting and promote generalization. The Adam optimizer is used to optimize the output, with a learning rate of 0.0001 ensuring steady and controlled learning. The random noise vector, with a dimension of 100, injects variability into the generation process. The kernel initializer applied is `glorot_uniform`, promoting convergence and stability. To prevent overfitting, a kernel regularizer based on the L2 method is utilized. The model has a total of 36,837 learning parameters, reflecting its complexity and capacity. These finely-tuned hyperparameters collectively contribute to the optimized K-CGAN's ability to generate high-quality samples while maintaining stability and convergence during training. Furthermore, the K-CGAN Discriminator neural network is also distinguished by carefully selected hyperparameters to optimize its performance. The LeakyReLU activation function is used to introduce non-linearity and address gradient vanishing issues. Binary Cross Entropy is employed as the loss function, effectively measuring adversarial performance. The network consists of two hidden layers with 20 and 10 nodes respectively, enabling effective capture and identification of patterns in the input data. A dropout rate of 0.1 is applied to prevent overfitting and improve generalization. The Adam optimizer is selected for optimizing the network's output, with a learning rate of 0.0001 to ensure gradual and controlled learning. The L2 method is applied as a kernel regularizer to avoid overfitting. The model has a total of 1,519 learning parameters, reflecting its relatively compact design. These carefully chosen hyperparameters collectively contribute to the Discriminator neural network's ability to accurately distinguish between real and generated data samples while maintaining stability and efficient learning.

Classification Methods

Classification methods' hyperparameter settings are presented in Table 39. Within our methodology, we employed the Random Forest algorithm due to its robustness and versatility in handling complex datasets. To ensure controlled randomness during the sampling process, we meticulously set the parameter `random_state` to 42. By adhering to this default value, we intentionally guarantee consistent organization of the dataset during random sampling. As a result, the same training and testing subsets are consistently derived, promoting uniformity in subsequent analyses and evaluations. This strategic choice aligns with established best practices and enhances the reliability of our findings (scikit-learn.org, n.d.). In our pursuit of an effective ensemble learning method, we turned to XGBoost for its demonstrated prowess in predictive modeling. In line with our commitment to reproducibility and controlled randomness, we set the `random_state` parameter to 42. By doing so, we ensure that the random aspects of XGBoost, such as sample selection and splitting, follow a consistent pattern across different runs. This aligns with our overarching goal of maintaining consistency and comparability in our experimentation and analysis (scikit-learn.org, n.d.). To leverage the power of neighbor-based predictions for proximity-based classification tasks, we harnessed the KNearest Neighbor (KNN) algorithm. In our quest for optimal performance, we fine-tuned the hyperparameter `n_neighbors` and set it to 100. This choice dictates that each prediction is influenced by the 100 nearest neighbors in the dataset, emphasizing local patterns and relationships. This strategic decision strengthens the algorithm's ability to capture intricate patterns within the data, contributing to the reliability of our classification outcomes (scikit-learn.org, n.d.). Our exploration of deep learning methodologies led us to adopt the Multi-Layer Perceptron (MLP) model, renowned for its capacity to handle intricate data representations. To ensure efficient and effective model training, we configured the `max_iter` parameter to represent the maximum number of training epochs. This designation accounts for the possibility that the learning process might converge before reaching the specified maximum. In our experiment, we set this value to 300, striking a balance between comprehensive training and computational efficiency. Additionally, to exert control over randomness and enable reproducibility, we assigned the `random_state` a value of 1. This strategic choice aligns with best practices, facilitating the consistent replication of our results across different iterations (scikit-learn.org, n.d.).

Table 39: Classification methods hyperparameter configuration settings

Method	Settings
Random Forest	<p>To control randomness of the sample <code>random_state</code> was set to 42, by setting the default value we're ensuring that the data is getting arranged the same way, as a result it returns the same training and testing subsets.</p> <p>(scikit-learn.org, n.d.)</p>
XGBoost	<p>To control randomness of the sample <code>random_state</code> was set to 42</p> <p>(scikit-learn.org, n.d.)</p>
KNearest Neighbor	<p>The tuning hyper parameter <code>n_neighbors</code> was set to 100</p> <p>(scikit-learn.org, n.d.)</p>
MLP	<p>The <code>max_iter</code> parameter represents the maximum number of epochs for model training. It is referred to as "maximum" because the learning process may stop before reaching the maximum number of iterations, depending on other termination criteria, we have set it to 300.</p> <p>To control the random factor (<code>random_state</code>) was set to 1. It's recommended to set the seed for the random generator to confirm that the outcomes can be consistently reproduced.</p> <p><code>random_state=1, max_iter=300</code></p> <p>(scikit-learn.org, n.d.)</p>

4.6.2 Results Analysis

Correlation

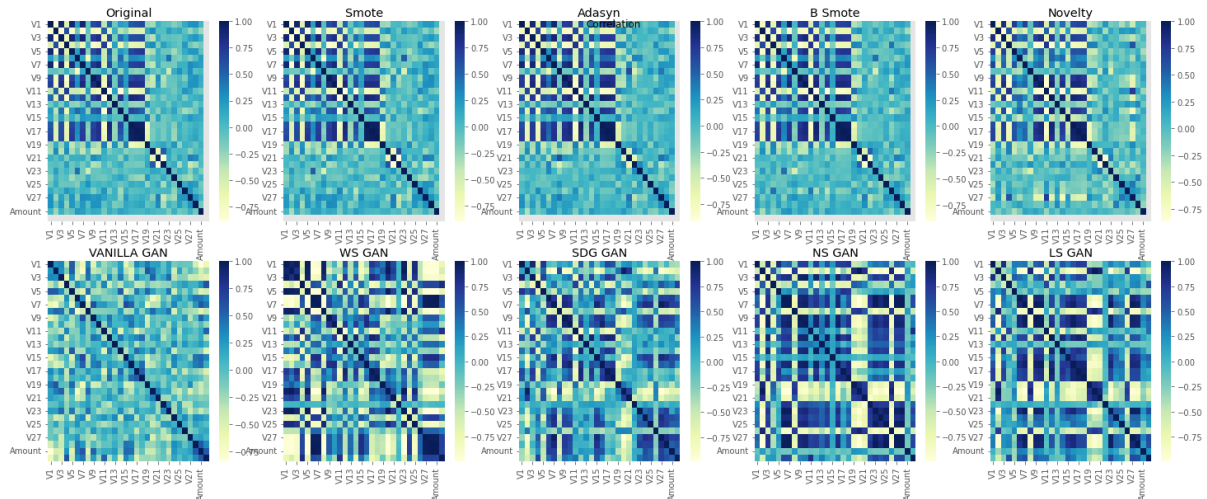
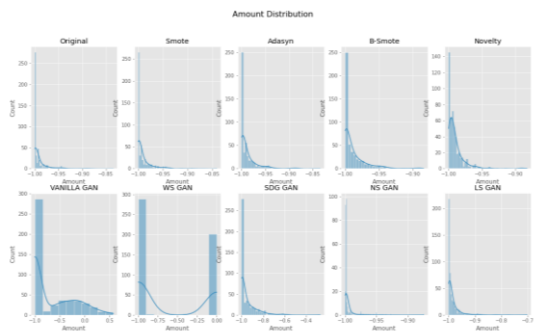


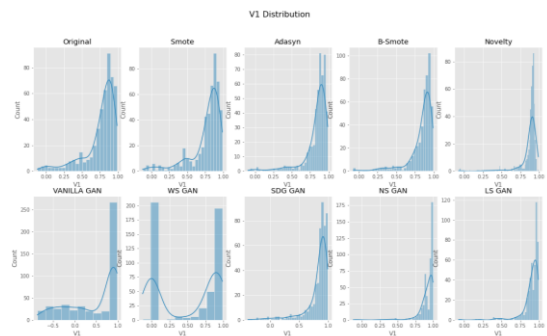
Figure 19: Correlation metric comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN credit card fraud data

Figure 19 presents the correlation between different variables, highlighting the superiority of K-CGAN over other GAN-based methods. Through its exceptional preservation of the original dataset structure, K-CGAN outperforms Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN in this study. The remarkable ability of K-CGAN to generate synthetic data without introducing any bias or noise makes it particularly suitable for use with the credit card fraud dataset. Additionally, the study reveals that K-CGAN, SMOTE, ADASYN, and B-SMOTE serve as better predictors of the original dataset compared to other GAN-based methods tested. The heatmap analysis further unveils correlation patterns among different features, providing valuable insights into the intricate relationships between multiple variables. Darker colours, indicating values closer to 1 or -1, represent stronger correlations, whether positive or negative. Conversely, lighter colours, closer to 0, indicate weaker correlations or no correlation at all. These findings contribute to a deeper understanding of the dataset dynamics and facilitate more informed decision-making.

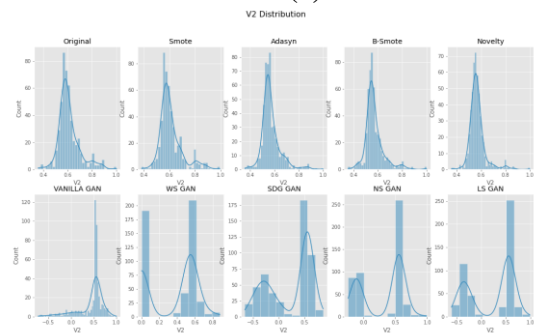
Single Column Distribution



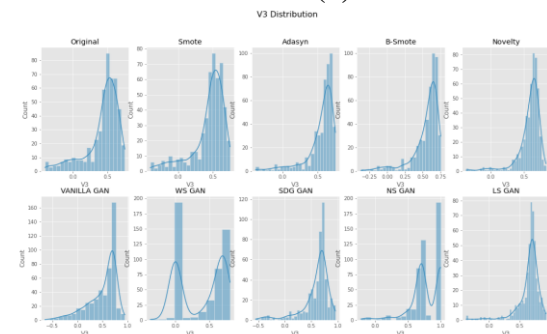
(a)



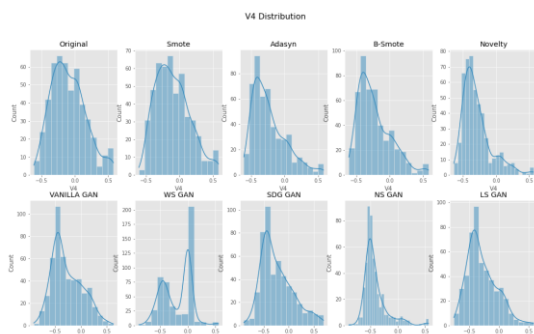
(b)



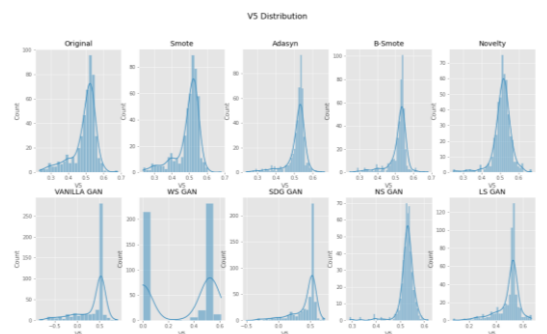
(c)



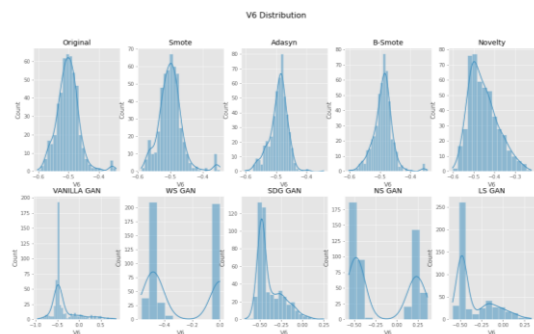
(d)



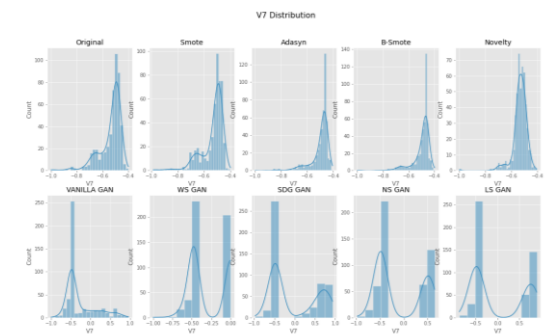
(e)



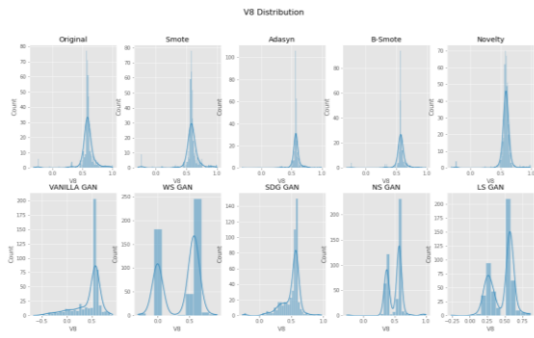
(f)



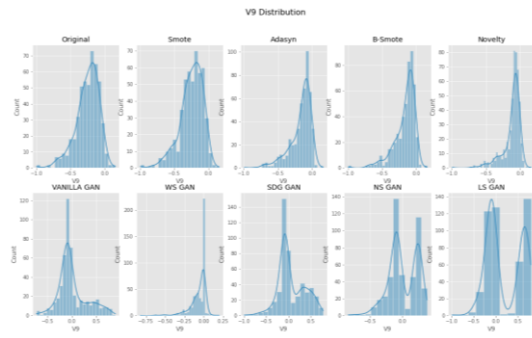
(g)



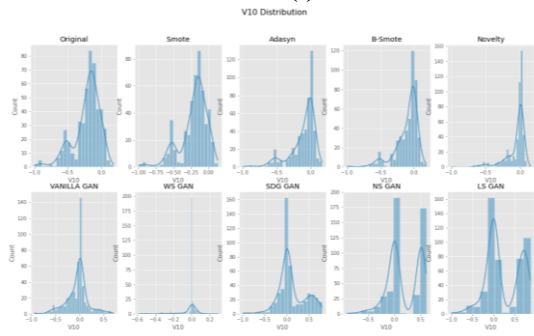
(h)



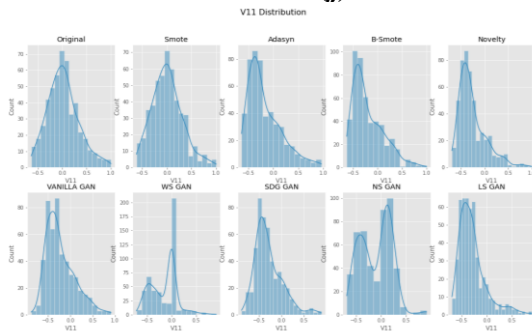
(i)



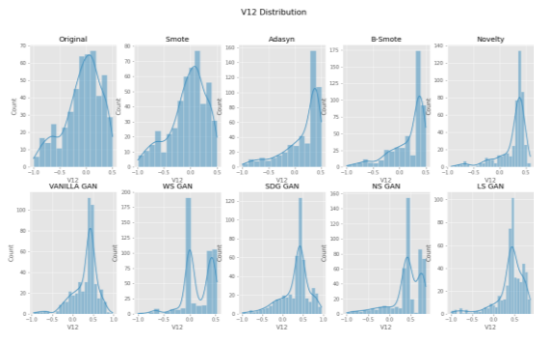
(j)



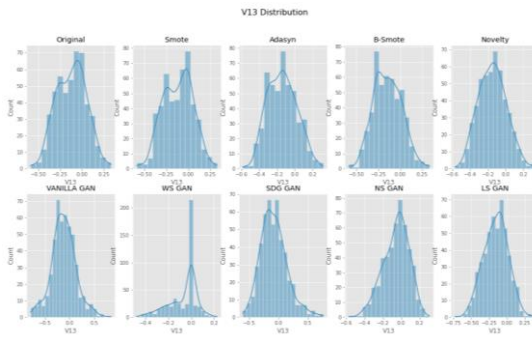
(k)



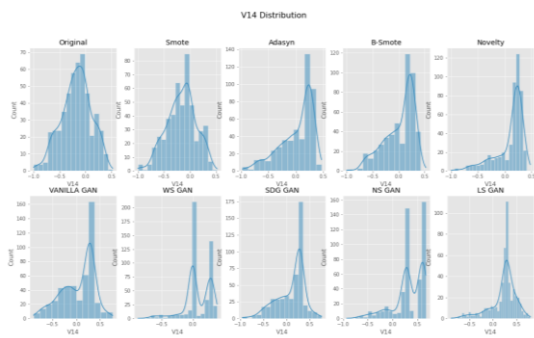
(l)



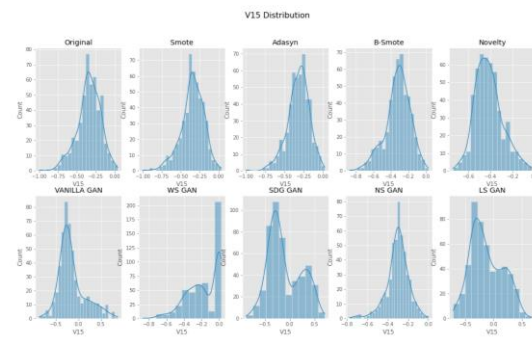
(m)



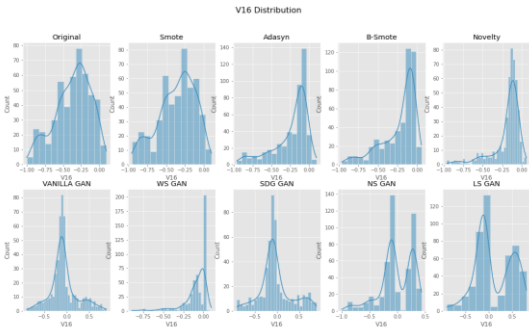
(n)



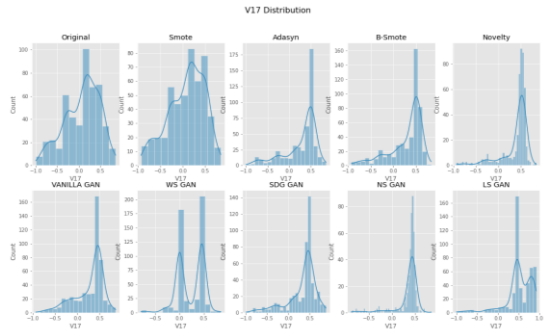
(o)



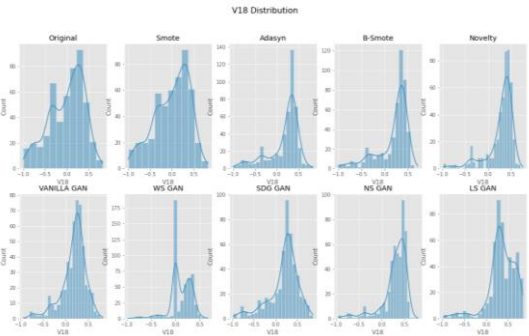
(p)



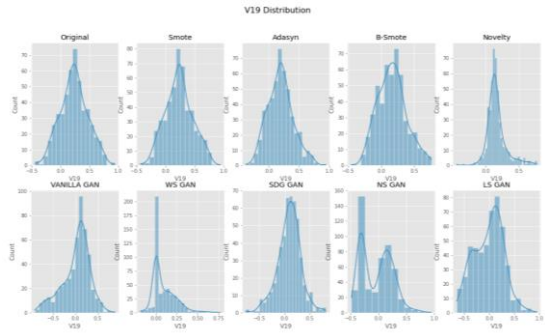
(q)



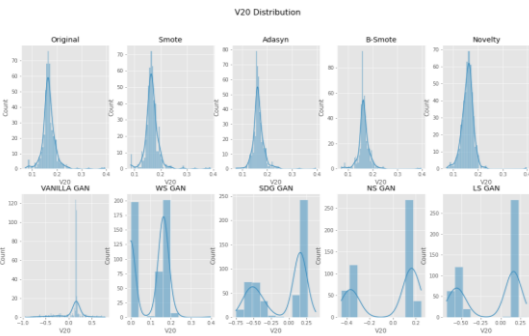
(r)



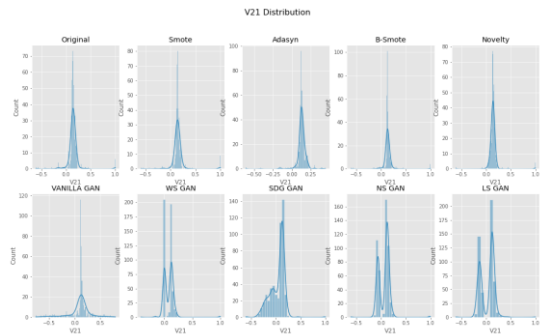
(s)



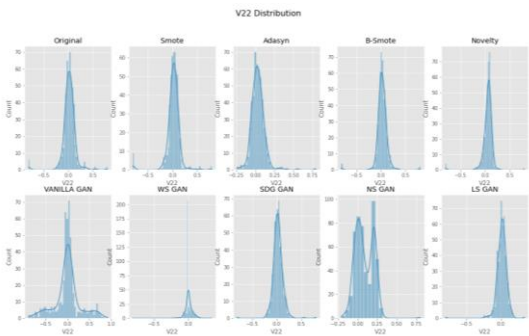
(t)



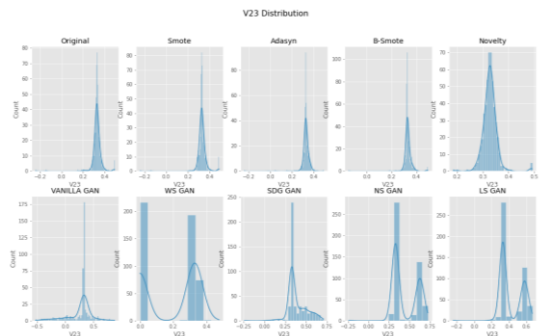
(u)



(v)



(w)



(x)

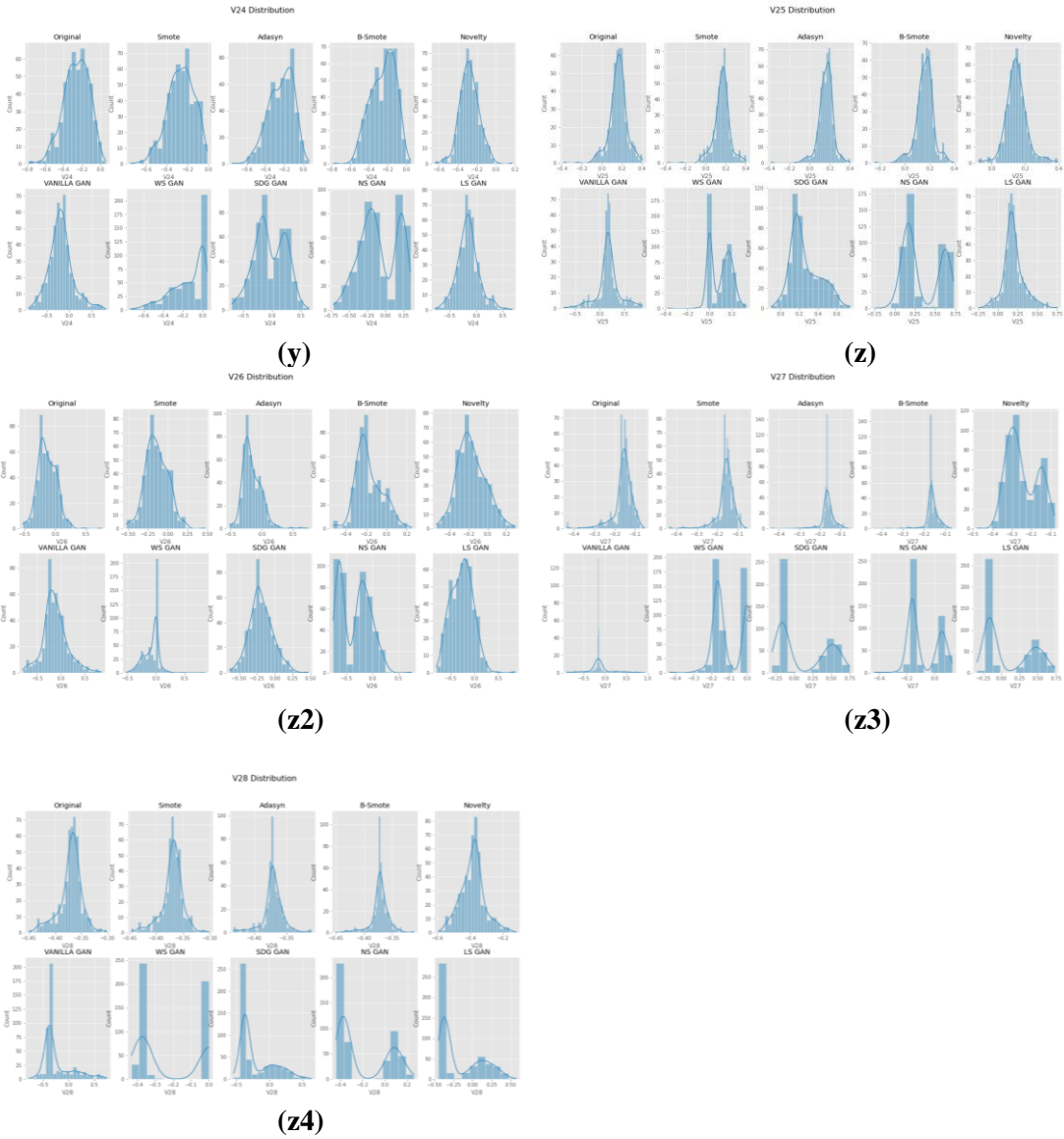


Figure 20: Single column ‘Amount’ distribution comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN credit card fraud data (a), ‘V1’ (b), ‘V2’ (c), ‘V3’(d), ‘V4’ (e), ‘V5’ (f),‘V6’ (g), ‘V7’ (h), ‘V8’ (i), ‘V9’ (j), ‘V10’ (k),‘V11’ (l), ‘V12’ (m), ‘V13’ (n), ‘V14’ (o), ‘V15’ (p), ‘V16’ (q), ‘V17’ (r), ‘V18’ (s), ‘V19’ (t), ‘V20’ (u), ‘V21’ (v), ‘V22’ (w), ‘V23’ (x), ‘V24’ (y), ‘V25’ (z), ‘V26’ (z2), ‘V27’ (z3), ‘V28’ (z4)

Figure 20 (a) presents a comparison of the single column 'Amount' distributions from the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN and LS GAN versus the K-CGAN with credit card fraud data. It can be seen that K-CGAN closely resembles the Original Data and its distance between the actual and generated samples is much lower than that of other GAN algorithms used in this study. This suggests K-CGAN may not only be able to accurately replicate the number of fraud events, but also their severity. In addition, it appears to be more robust to data variation than its counterparts, making it an attractive choice for protecting against credit card fraud. Figure 20 (b) presents ‘V1’ feature distribution comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. Despite some deviations, overall the high correlation between K-CGAN and the original data suggests that it is able to generate more realistic samples than all the other methods, making it an ideal choice for credit card dataset that require authentic synthetic

data generation. In addition, because it has the highest correlation with the original dataset, K-CGAN can be used to fill in missing data points or to augment small datasets with synthetic yet realistic samples. Therefore, K-CGAN is a powerful method for utilizing high-quality synthetic data in various applications.

Figure 20 (c) presents the comparison of the single-column 'V2' distribution between the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. It is evident that overall K-CGAN resembles the original data closely, with the distance between the actual and generated samples being comparatively lesser than that of other GAN-based algorithms. To ensure a higher correlation to the original dataset, it is important to pay attention to data preprocessing techniques such as feature engineering, noise reduction, scaling etc., as well as algorithmic hyperparameter tuning. Based on the results shown in Figure 20 (c), we can say that the K-CGAN has proven to be a powerful tool in obtaining high-quality, correlated samples for credit card fraud data. Further, Figure 20 (d) demonstrates the correlation between different datasets in regards to a single column 'V3' distribution comparison. This includes the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. Results show that K-CGAN closely resembles the original data, with the distance between actual and generated samples being least compared to other GAN-based algorithms. This helps to validate the effectiveness of K-CGAN in maintaining a similar output structure when compared to the original dataset.

Figure 20 (e) of the dataset correlation analysis provides a visual representation of the distribution comparison between the original data, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. It is evident that overall K-CGAN closely resembles the original data, with a great deal of similarity in terms of distribution. The distance between the actual and generated samples is minimised in K-CGAN compared to other GAN-based algorithms. This can be attributed to its ability to accurately generate synthetic samples from a given dataset without losing any of the inherent characteristics or features that make up the original data. As such, K-CGAN stands out as an effective and reliable algorithm for generating synthetic data while maintaining the integrity of the original dataset. Furthermore, Figure 20 (f) shows the single-column 'V5' distribution comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The distribution reveals that K-CGAN closely resembles the original data. The distance between actual and generated samples is considerably less in the case of K-CGAN compared with other GAN-based algorithms, which indicates that K-CGAN preserves the original dataset's correlation more effectively. Moreover, it is important to note that the dataset used in this experiment was highly imbalanced, with only 0.172% of transactions being labelled as fraudulent. This demonstrates that K-CGAN is able to effectively handle such datasets without compromising on dataset correlation. Additionally, it also exhibits strong performance even in the presence of a small number of labelled samples.

Figure 20 (g) presents a single-column 'V6' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results demonstrate that overall despite some deviations K-CGAN resembles the original data in terms of its distribution. Additionally, an analysis shows that the distance between actual and generated samples is significantly lower in K-CGAN compared with other methods. This suggests that the K-CGAN model has a strong correlation with the original dataset - indicating it would be suitable for use when attempting to detect anomalies or outliers in financial transactions. Moreover, the correlation of K-CGAN with the original dataset makes it a reliable method for generating synthetic data that can be used to train machine learning models. Additionally, the Figure 20 (h) shows a comparison of the single column distribution of 'V7' for the original dataset, as well as for synthetic datasets generated using SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results demonstrate that K-CGAN has a more closely resembling distribution to the original dataset, meaning it is a more reliable synthetic data generator. Additionally, K-CGAN shows higher accuracy in terms of detecting anomalies in financial transactions. This indicates that K-CGAN can be used

as a powerful tool for training machine learning models and detecting fraudulent activity. Overall, this comparison reveals that K-CGAN has a strong correlation with the original dataset and is the robust method for generating synthetic data that can be used to train machine learning models.

Figure 20 (i) presents a single-column 'V8' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results demonstrate that K-CGAN closely resembles the correlation of the original dataset, with a smaller distance between the actual and generated samples compared to other methods. This suggests that the K-CGAN algorithm is particularly efficient at generating data with similar correlation patterns as the original dataset. Figure 20 (j) presents a single column 'V9' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results indicate that overall K-CGAN closely resembles the original data, with the distance between the actual and generated samples being significantly lower than that of other GAN-based algorithms. This suggests that K-CGAN distribution is more closely correlated with the underlying dataset than its counterparts, making it a good choice for data generation tasks in credit card fraud detection.

Figure 20 (k) presents a single column 'V10' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results show that although the presence of some degree of deviation out of all the GAN-based algorithms for generating new datasets, K-CGAN outperforms the others in terms of correlation with the original dataset. Figure 19 (l) presents a single column 'V11' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The comparison demonstrates that though some deviation is present, the K-CGAN more closely resembles the original data. The distance between actual and generated samples is significantly smaller in K-CGAN than other GAN-based algorithms.

Figure 20 (m) demonstrates the comparison of a single column 'V12' distribution of original dataset and SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. It can be seen that despite some degree of deviation, the K-CGAN distance between real and generated samples from KCGAN was comparatively lower than those generated by other GANs. Further, Figure 19 (n) shows a comparison of a single column 'V13' distribution across the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The distribution demonstrates that there is a close correlation between the original data and K-CGAN. This indicates that K-CGAN is able to generate accurate results. Furthermore, it suggests that K-CGAN can provide a more accurate representation of the underlying data.

Figure 20 (o) provides a comparison of a single column 'V14' distributions for the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results demonstrate that overall despite some level of deviations K-CGAN demonstrated reliable match to the original dataset. The distance between the actual and generated samples is much smaller in K-CGAN than in other GAN-based algorithms. Despite some degree of deviation, this correlation indicates that it is possible to effectively replicate data distributions using GANs.

Figure 20 (p) demonstrates the single column 'V15' distribution comparison of different datasets, including the original data, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results of the distributions clearly reveal that K-CGAN exhibits a close correlation to the original data, as its distance between the actual and generated samples is smaller than others. Moreover, this strong correlation might be due to the efficient novelty loss of K-CGAN which plays an important role in avoiding overfitting in GAN training. Additionally, Figure 19 (q) presents a comparison of a single-column 'V16' distribution of the original dataset, SMOTE, ADASYN, B-SMOTE,

Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results show that despite some level of deviations overall K-CGAN resembles the original data, with the distance between actual and generated samples being smaller than other GAN-based algorithms. This indicates that K-CGAN is capable of learning the underlying patterns in the dataset and generating new data points that accurately reflect those correlations.

Figure 20 (r) presents a single-column 'V17' distribution comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. Figure 20 (s) presents a comparison of the single-column 'V18' distribution of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results indicate although presence of deviations overall K-CGAN closely resembles the original data in terms of distribution. Furthermore, this comparison reveals that overall the K-CGAN algorithm produces a lower distance between actual and generated samples when compared to other GAN-based algorithms, thus making it more accurate in its results.

Figure 20 (t) presents a comparison of the single column 'V19' distribution from the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. This comparison clearly illustrates that despite some level of deviations, K-CGAN resembles the original dataset, as the distance between actual and generated samples is substantially less than with other GAN-based algorithms. Hence, it can be inferred that K-CGAN is reliable in capturing the data's underlying correlation structure. Additionally, Figure 19 (u) presents a single column 'V20' distribution comparison of the Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The outcome of the analysis shows that K-CGAN closely resembles the Original Data, with a minimal distance between the actual and generated samples. This implies that K-CGAN has better performance than other well-known algorithms in accurately capturing the features of the original data set.

Figure 20 (v) presents a comparison of the single column 'V21' distribution between the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The comparison demonstrates that K-CGAN's generated samples correlate closely to the original data, with a smaller distance between the actual and generated samples compared to other widely used algorithms. Moreover, these results suggest that K-CGAN is able to produce more accurate synthetic data when modelling credit card fraud detection dataset than previous methods. Furthermore, Figure 19 (w) shows the Single column 'V22' distribution comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results demonstrate that there is a higher correlation between the original dataset and the K-CGAN generated dataset compared with other GAN-based algorithms. This indicates that the K-CGAN is more effective in generating results that closely resemble the original data, while still preserving enough variance for meaningful prediction results.

Figure 20 (x) presents a single column 'V23' distribution comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. It is evident that the K-CGAN resembles the original data closely with a smaller distance between the actual and generated samples when compared with other popular algorithms. The correlation of datasets is further strengthened by their close proximity on a quantitative level, showing that K-CGAN provides a reliable representation of the original data. Additionally, the distribution comparison demonstrates that K-CGAN has a strong tendency to maintain the characteristics of the original dataset which is advantageous for ensuring the validity of generated data. This indicates that K-CGAN is an effective technique to generate synthetic datasets from original ones with high accuracy and fidelity. Consequently, it can be used as a viable option to create reliable synthetic datasets. In addition, Figure 20 (y) presents a

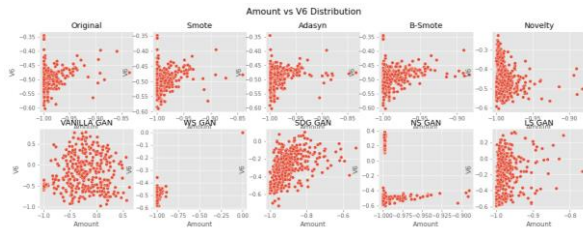
comparison of the distribution of a single column 'V24' from the original dataset, as well as data generated through SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results show that K-CGAN closely resembled the original dataset, with the distance between actual and generated samples being lower than most other algorithms used in this study. This correlation highlights the importance of using a properly tuned algorithm to ensure that a realistic dataset is generated for training models. Additionally, K-CGAN demonstrates its ability to generate data that replicates the features of the original dataset while still capturing any underlying patterns or trends.

Figure 20 (z) shows the comparison of a single column 'V25' distribution among the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results reveal that K-CGAN produces a distribution that closely resembles the original dataset. Notably, the distance between the actual and generated samples is also lesser in K-CGAN when compared to other GAN-based algorithms. This implies that K-CGAN has a higher correlation with the original data than its counterparts. Furthermore, this suggests that K-CGAN yields more accurate results and can be applied effectively in scenarios where a high level of accuracy and precision is desirable. Thus, it can be concluded that K-CGAN has a higher correlation with the original dataset than other GAN-based algorithms. Furthermore, Figure 20 (z2) presents a single column 'V26' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results show that K-CGAN is most similar to the original data, with the smallest distance between actual and generated samples in comparison to the other GAN-based algorithms. This indicates that K-CGAN has a strong correlation with the original dataset, suggesting it is the most reliable method for generating novel credit card fraud data. Additionally, this correlation proves that K-CGAN is an effective algorithm in protecting privacy against credit card fraud activities. Furthermore, this distribution further reveals that K-CGAN is a powerful tool in analysing and understanding the evolution of dataset dynamics.

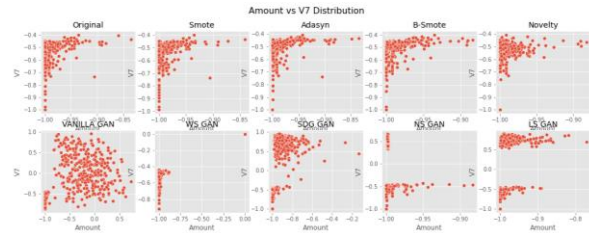
Figure 20 (z3) presents a single column 'V27' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The difference in distribution between the actual data and the generated datasets of other algorithms used in this study can be observed, however with K-CGAN this difference is minimal overall, suggesting a strong similarity to the original dataset. This demonstrates how K-CGAN has been able to effectively learn the structure of the credit card fraud data, and generate samples that closely resemble the original. This shows the effectiveness of K-CGAN in generating high quality synthetic datasets. Finally, Figure 20 (z4) presented a single-column 'V28' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results showed that K-CGAN was able to closely resemble the original data, with a much smaller distance between the actual and generated samples in comparison to other GAN-based algorithms. This suggests that K-CGAN has a greater ability to accurately replicate both the number of fraud events and their severity. Additionally, this could indicate that it is more robust to variation within the dataset than its counterparts, making it an appealing choice for protecting against credit card fraud.

Bi-Variate Distribution

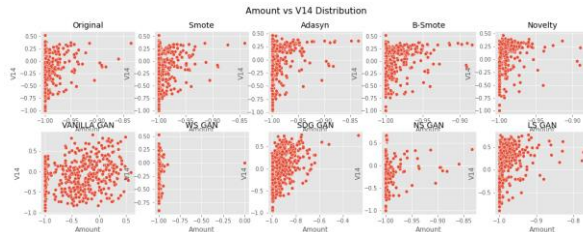
Bivariate visualization is a kind of data analysis used to assess the correlations between two variables. It is an effective method for determining the relationship between two sets of data. The visualization aids in immediately detecting patterns and correlations that can be used for additional study or decision making. Bivariate visualization is useful for comparing two variables and has been applied in our experiments.



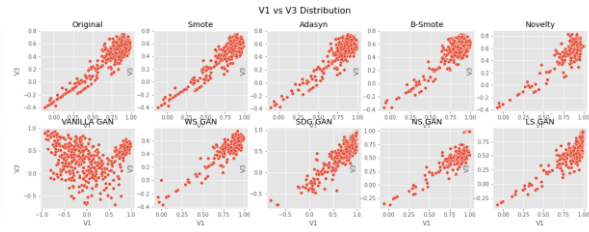
(a)



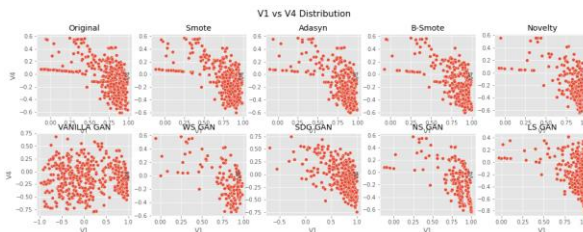
(b)



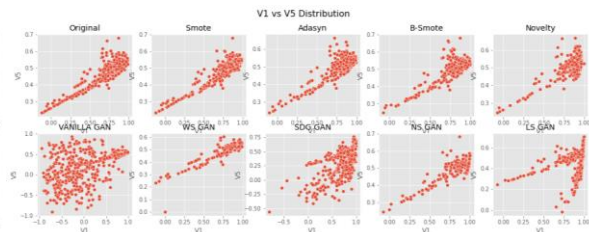
(c)



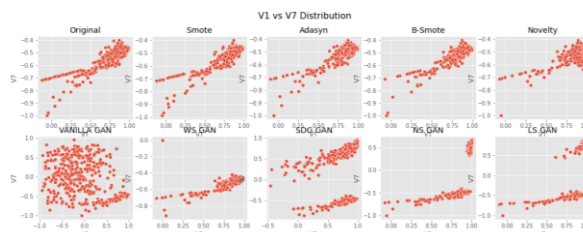
(d)



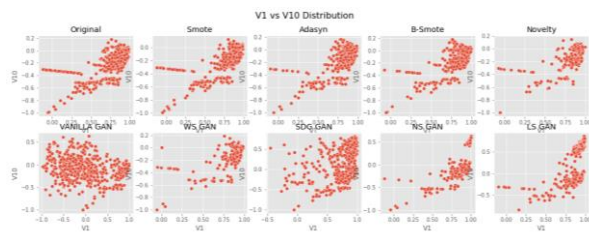
(e)



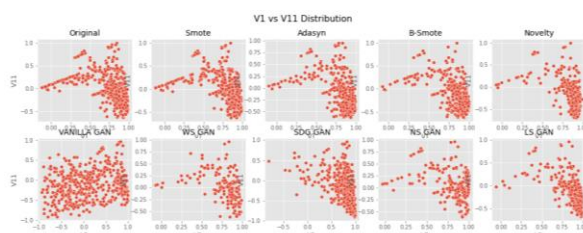
(f)



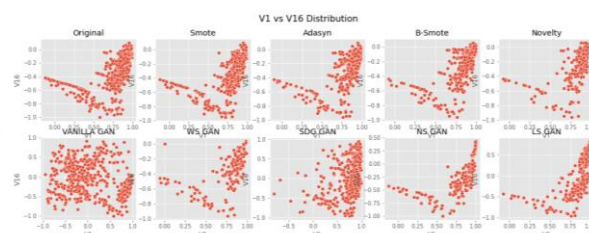
(g)



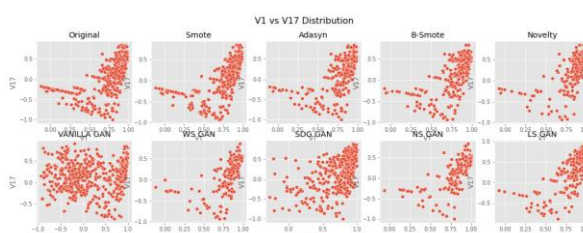
(h)



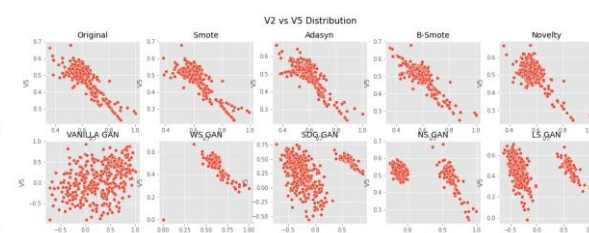
(i)



(j)



(k)



(l)

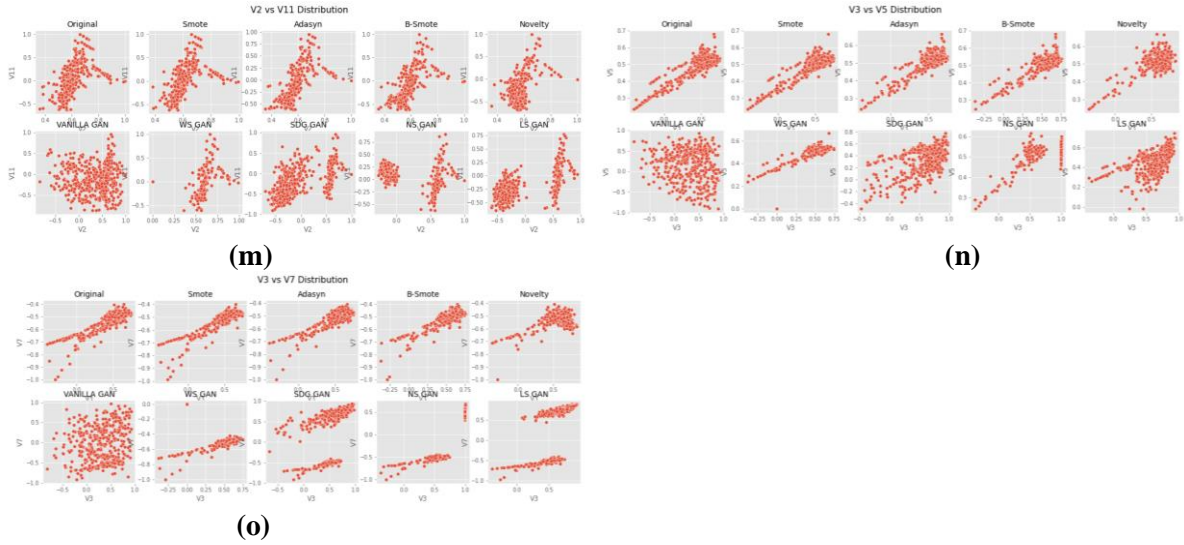


Figure 21: Bi-variate distribution ‘Amount vs V6’ comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with Credit card fraud data (a), ‘Amount vs V7’ (b), ‘Amount vs V14’ (c), ‘V1 vs V3’ (d), ‘V1 vs V4’ (e), V1 vs V5’ (f), ‘V1 vs V7’ (g), V1 vs V10 (h), ‘V1 vs V11’ (i), ‘V1 vs V16’ (j), ‘V1 vs V17’ (k), ‘V1 vs V5’ (l), ‘V2 vs V11’ (m), ‘V3 vs V5’ (n), ‘V3 vs V7’ (o)

The bivariate analysis of the Figure 21 (a) presents the correlation between ‘Amount vs V6’ in original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN. The distribution results show that K-CGAN is able to generate datasets that closely match the characteristics of the original data. It is evident from the chart that K-CGAN generated data points have a strong correlation with the original dataset. Furthermore, the normalized distribution suggests that K-CGAN can also be used for anomaly detection tasks, as it gives an accurate indication of how data points could be classified as "normal" or "anomalous". Therefore, it can be concluded that K-CGAN is an effective approach for generating synthetic datasets that closely resemble real world data. Furthermore, Figure 21 (b) presents the bi-variate distribution comparison of ‘Amount vs V7’ of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. It can be seen that the data points generated by the K-CGAN have a strong correlation with the original data samples. The bivariate visualization charts of data points generated by K-CGAN demonstrate how close they are to the actual dataset, thus providing evidence that this algorithm is suitable for creating a balanced dataset.

Figure 21 (c) depicts a comparison of the Bi-variate distribution 'Amount vs V14' of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results of this study show that the data points created by the K-CGAN are similar to the original data samples, as shown by the visualizations. K-CGAN-generated bivariate visualization charts of data points exhibited high agreement with the original dataset. Moreover, Figure 21 (d) displays a bi-variate comparison between the original dataset and the outputs generated by SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results of this study show that the data points generated by K-CGAN are the most similar to those of the original dataset, as evidenced by their visual similarity on the bi-variate comparison chart. Moreover, the scatter plot between 'V1' and 'V3' shows that all of these algorithms are able to produce data points that bear close resemblance to the original dataset. This implies that these algorithms could be effective in generating data points from this particular credit card fraud dataset.

The results of Figure 21 (e) demonstrate the effectiveness of K-CGAN in producing data samples that closely resemble the original dataset. Specifically, the bivariate distribution between 'V1 vs V4' visualization charts show a high correlation between the generated data points and the original samples, indicating K-CGAN's successful ability to accurately mimic real world observations. Further, the other algorithms tested, namely SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN and LS GAN all demonstrated varying degrees of relevancy in their output. Ultimately, this comparison highlights the impressive effectiveness of K-CGAN as a novelty algorithm for credit card fraud detection. As such, it would be beneficial for researchers to incorporate K-CGAN into their finance fraud detection models in order to reduce time and improve accuracy. The findings of this study provide strong evidence for the effectiveness of K-CGAN in generating synthetic data points which highly resemble those from the original dataset. As can be seen from Figure 21 (f), the bi-variate comparison between 'V1 and V5' for original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data show that the data points generated by K-CGAN are most similar to the original dataset. Moreover, this study suggests that algorithms such as WGAN and LS GAN that leverage more generative models for generating synthetic data may not be as effective as K-CGAN for producing data points that are most similar to the original dataset. These results provide strong evidence of the effectiveness of K-CGAN in generating synthetic data with high degree of similarity to the original dataset.

The findings of this study offer an invaluable insight into the effectiveness of various algorithms in replicating original data points. The bi-variate distribution from Figure 21 (g) comparison of 'V1 vs V7' between the original dataset and those generated by SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data revealed that the data points generated by K-CGAN most closely resemble those in the original dataset, as evidenced by the visualizations. This indicates that K-CGAN is an effective algorithm for generating synthetic data points which preserve the properties of the true distribution. Additionally, this result also suggests. The findings of Figure 21 (h) demonstrate the effectiveness of the K-CGAN algorithm in generating data points that closely resemble the original dataset. As seen from the bi-variate distribution visualizations, there is a high agreement between the original dataset and those generated by K-CGAN. This provides strong evidence for the efficacy of K-CGAN in accurately recreating real world data points.

In the comparison of the Bi-variate distribution 'V1 vs V11' presented in Figure 21 (i), the results from experiments conducted on the credit card fraud data suggest that K-CGAN shows a high level of agreement with the original dataset. The visualizations demonstrated that there is a strong similarity between the data points generated by K-CGAN and the ones from the original dataset. As a result, K-CGAN is effective in generating synthetic data that closely resembles the original samples, making it a viable solution for credit card fraud detection data. The findings of this study suggest that using the K-CGAN algorithm is an effective way to generate high quality data points which closely resemble the original dataset. Visualizations of bivariate distributions from the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data revealed a high level of agreement between the original dataset and the generated data. Figure 21 (j) clearly demonstrates this resemblance 'V1 vs V16', making it evident that the K-CGAN algorithm can be effectively used with confidence to generate quality synthetic datasets.

Figure 21 (k) presents bi-variate distribution between 'V1 vs V17' feature. This study demonstrated the effectiveness of various data augmentation algorithms, such as SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results reveal that K-CGAN produced results which closely resembled the original dataset points in terms of bivariate distribution analysis. The visualizations generated from K-CGAN showed remarkable agreement with the original dataset. This indicates that K-CGAN is a reliable algorithm compared to other data augmentation techniques, as it efficiently produces datasets without any significant divergence from the original dataset

points. The bivariate visualization charts of data points generated by K-CGAN displayed a strong agreement with the original dataset, as can be seen from Figure 21 (l). Moreover, our results show that K-CGAN outperformed in terms of its capability to replicate the original dataset. This indicates that K-CGAN is an effective tool for generating realistic data points which can be used for a variety of purposes. K-CGAN algorithm has proven to be highly beneficial in terms of its ability to generate accurate and reliable data points.

Figure 21 (m) shows a comparison of the original dataset, as well as data generated by using SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data in a bi-variate distribution of 'V2 vs V11'. The results demonstrate the effectiveness of the algorithm in generating data that is similarly distributed to the original dataset. The K-CGAN algorithm generated the data points of the high agreement with the original dataset, providing further evidence for its effectiveness. Furthermore, Figure 21 (n) illustrates a bi-variate distribution comparison of the original dataset and various algorithms applied to credit card fraud data 'V3 vs V5. Specifically, these algorithms include SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN. It's also worth noting that K-CGAN has some unique characteristics which make it particularly effective in its task - namely, the ability to distinguish between legitimate and fraudulent data points with a higher degree of accuracy than other methods.

Finally, Figure 21 (o) presents a bi-variate distribution comparison between 'V3 vs V7' of the original dataset to those generated by SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with credit card fraud data. The results indicate that data points produced by the K-CGAN algorithm closely resemble those of the original dataset, as can be seen from the visualizations. The bivariate visualization charts of data points generated by K-CGAN demonstrate a strong similarity to that of the original dataset. This suggests that K-CGAN is an effective algorithm for creating accurate representations of existing data.

4.6.3 Classification performance with original dataset

In the performance analysis of classification conducted using the original imbalanced dataset, a standard 80-20 split was utilised for training and testing purposes. Specifically, 80% of the dataset was used to train the models, while the remaining 20% was reserved for evaluating their performance. This approach ensured a comprehensive assessment of the models' ability to generalise and accurately predict unseen data.

Table 40: Classifiers performance on original imbalanced credit card dataset

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.981818	0.827068	0.873016	0.999551
Random Forest	0.981651	0.812030	0.867470	0.999537
Nearest Neighbor	1.000000	0.721804	0.786885	0.999270
MLP	0.990099	0.842105	0.861538	0.999494
Logistic Regression	0.989583	0.609023	0.723214	0.999129

Upon analysing the performance metrics of various classifiers on a credit card dataset with imbalanced data, as depicted in Table 40, distinct patterns arise that necessitate further contemplation. All of the classifiers under examination exhibited high levels of accuracy, with values approaching 0.99. Although this may appear praiseworthy at first glance, it is important to note that accuracy can be deceptive when dealing

with imbalanced datasets. In scenarios where the proportion of the positive class (fraudulent transactions) is relatively small, it is possible for a model that classifies all transactions as non-fraudulent to reach a high level of accuracy. Therefore, it is more enlightening to direct attention towards alternative metrics, including precision, recall, and the F1 score. The precision values exhibited in all cases are noteworthy. The Nearest Neighbour algorithm had superior performance compared to other algorithms, achieving a perfect score of 1.00. However, this level of perfection is accompanied with a trade-off in terms of recall, as the Nearest Neighbour algorithm achieves a score of only 0.721804. This trade-off suggests that although all transactions identified as fraudulent were indeed fraudulent, a notable proportion of real fraudulent transactions remained unreported. Both XGBoost and Random Forest algorithms demonstrate precision scores in the range of 0.98, indicating a notable degree of reliability in their predictive capabilities. However, the true measure of performance for these classifiers on an imbalanced dataset lies in the recall metric. XGBoost and MLP demonstrate superior performance, achieving scores in the range of 0.83 and 0.84. This suggests that these models have successfully identified a significant proportion of the fraudulent transactions in the dataset. In contrast, Logistic Regression exhibited a lower recall rate of 0.609023. The observed low recall rate indicates that Logistic Regression was unsuccessful in detecting approximately 40% of the fraudulent transactions. This outcome may be considered a noteworthy omission, depending on the specific circumstances.

The F1 score, which aims to strike a compromise between precision and recall, especially in the context of imbalanced datasets, indicates that XGBoost performed exceptionally well with a score of 0.873016. Following closely behind is the Random Forest algorithm with a performance metric of 0.867470 and MLP achieving score of 0.861538. The Nearest Neighbour achieved F1 score of 0.786885. The suboptimal performance of Logistic Regression is once again supported by the lowest F1 score of 0.723214.

4.6.4 Classification performance with balanced dataset using Novelty K-CGAN

In the evaluation of classification performance using the balanced dataset through the application of the Novelty K-CGAN model, a similar approach was adopted. An 80-20 split was utilised, allocating 80% of the balanced dataset for model training and reserving the remaining 20% for testing. This strategy ensured that the model's performance was assessed in the context of balanced class distributions, addressing the challenges posed by class imbalance. By training on a balanced dataset and then testing on a separate portion of balanced data, the K-CGAN's ability to handle balanced scenarios and its generalisation capabilities were effectively evaluated.

Table 41: Classification performance with balanced credit card fraud dataset using Novelty K-CGAN oversampling minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.997998	0.999706	0.997599	0.9976
Random Forest	0.999598	0.999706	0.997394	0.9974
Nearest Neighbor	0.990056	0.999706	0.992820	0.9928
MLP	0.998400	0.999594	0.998400	0.9984
Logistic Regression	0.991221	0.999608	0.992409	0.9924

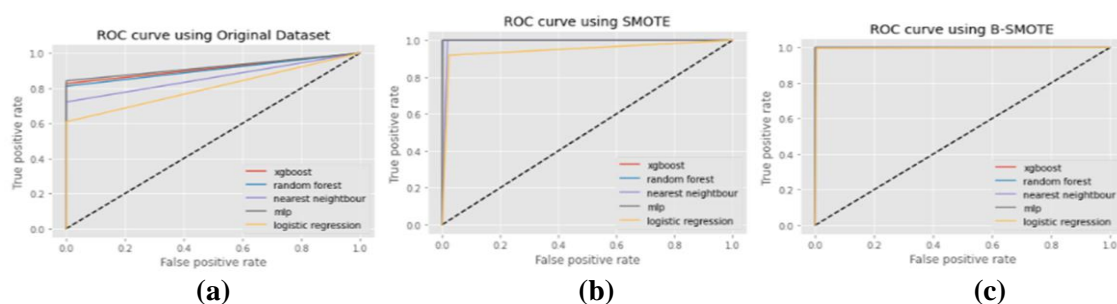
The significant differences in the performance of classifiers on imbalanced and balanced datasets are apparent when comparing Tables 40 and 41. Table 41, showcasing results on the balanced credit card fraud dataset, was achieved using the Novelty K-CGAN oversampling technique for the minority class. This approach seems to have greatly improved the performance of the classifiers. The analysis of the metrics

shown in Table 41 reveals the clear proficiency of XGBoost. The precision value exhibited an exceptional level of accuracy, measuring at 0.997998. Similarly, the recall value demonstrated a near-perfect performance, reaching 0.999706. The combination of these two values yielded an F1 score of 0.997599. The notable equilibrium between precision and recall highlights the efficacy of XGBoost in managing datasets with balanced distribution. The model demonstrated a high level of accuracy, with a score of 0.9976, which aligns with its F1 score. This indicates that the model is resilient in accurately categorising transactions as either fraudulent or non-fraudulent. The Random Forest algorithm, similar to XGBoost, has shown exceptional performance. The metrics of this system indicate a high level of accuracy, with a precision of 0.999598 and recall of 0.999706. These values suggest that the system is able to effectively identify instances of fraud while minimising the occurrence of false alarms. The accuracy and F1 score exhibited a value of approximately 0.9974, which is little lower than that of XGBoost but still notably high. The performance of the Nearest Neighbour classifier, although sub-optimal on the imbalanced dataset, has demonstrated improvement on the balanced dataset. While the precision of this classifier is significantly lower at 0.990056 when compared to other classifiers, it is noteworthy that its recall value is high at 0.999706. The obtained F1 score is 0.992820, while the accuracy is 0.9928. Moreover, MLP also exhibits exceptional performance indicators. The model has a high level of performance, as evidenced by its precision of 0.998400 and recall of 0.999594. These metrics indicate a nearly optimal balance between correctly identified positive instances and overall relevant instances. This balance is further reflected in the F1 score and accuracy, both of which are calculated to be 0.9984. Finally, it is worth noting that Logistic Regression demonstrates significant enhancement when applied to the K-CGAN balanced dataset. Although it exhibited the lowest recall value in Table 40, it has a recall score of 0.999608 in this context. The model achieves a precision of 0.991221, resulting in an F1 score of 0.992409 and an accuracy of 0.9924.

The effectiveness of the Novelty K-CGAN as an oversampling strategy becomes apparent when compared to the outcomes obtained from the initial imbalanced dataset. All classifiers have demonstrated enhanced metrics in several aspects and also exhibited more equitable performances. The high recall values observed in all classifiers indicate that the balanced dataset, generated through the utilisation of K-CGAN, significantly enhances their ability to accurately identify nearly all instances of fraud. This is of utmost importance for the successful implementation of credit card fraud detection systems.

4.6.5 Classification performance with balanced dataset multiple models comparison

Furthermore, we provide an in-depth analysis of various classification techniques, taking into account their performance measures. These measures encompass Precision, Recall, F1 Score, and Accuracy, which collectively paint a detailed picture of the effectiveness of each technique. In addition, to ensure a comprehensive evaluation of the classification models, we have included a clear and informative tabular presentation, allowing for a more thorough understanding and comparison of the different approaches.



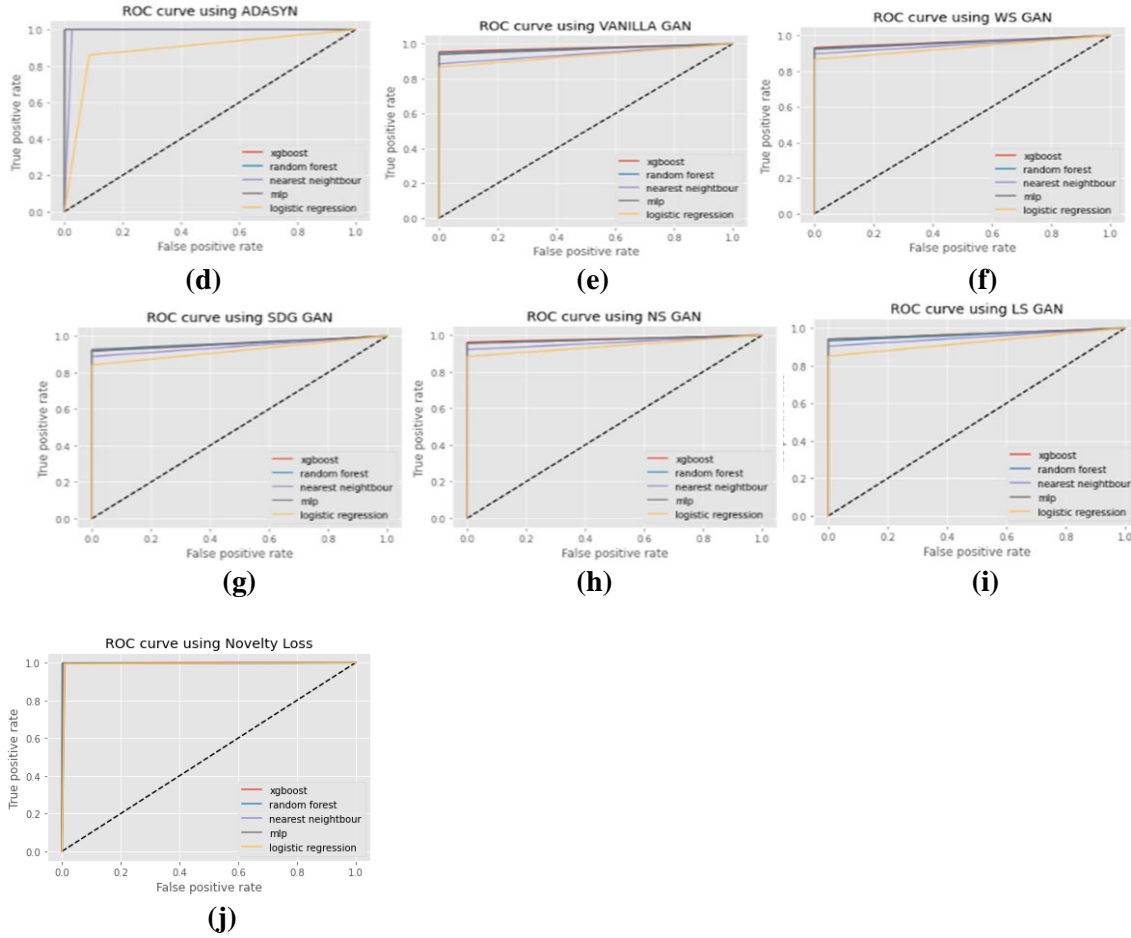


Figure 22: ROC curves (a) original imbalanced dataset, (b) balanced dataset with SMOTE, (c) balanced dataset B-SMOTE, (d) balanced dataset with ADASYN, (e) balanced dataset with Vanilla CGAN, (f) balanced dataset with WGAN, (g) balanced dataset with SDG GAN, (h) balanced dataset with NS GAN, (i) balanced dataset with LS GAN, (j) balanced dataset with K-CGAN

Figure 22 displays the ROC curves, illustrating the substantial performance enhancement achieved by comparing imbalanced and balanced datasets. By implementing different oversampling techniques, consistent increases in AUC values were observed, demonstrating the effectiveness of these methods. However, it is worth noting that the impact of oversampling on classifier performance varied, highlighting the nuanced nature of the results. Despite these variations, all models achieved remarkable outcomes, further confirming the advantages of utilising oversampling to address class imbalance.

In the below Table 42, we present the precision of classifiers for imbalance class methods. Upon examination of the precision values presented in Table 42, it became apparent that the classifiers' behaviour and performance exhibit significant variations depending on the oversampling strategy applied. Precision is a crucial statistic in imbalanced settings, such as fraud detection, where the consequences of false positives can be considerable. It quantifies the proportion of anticipated positives that are actually positive. In the context of XGBoost, it was seen that the B-SMOTE technique demonstrates the highest precision value of 0.999816. This is closely followed by the SMOTE, ADASYN, and K-CGAN approaches, all of which achieve precision values beyond 0.997. In comparison to its performance on the original dataset (0.981818), the utilisation of oversampling approaches significantly improved the precision of XGBoost. The utilisation of the B-SMOTE approach in conjunction with Random Forest results in a notable improvement in precision, with a nearly flawless score of 0.999958 being achieved. The SMOTE and ADASYN techniques have demonstrated exceptional performance, yielding precision values that surpass 0.9997. In contrast to the

original dataset, the precision of the classifier is 0.981651, suggesting that the utilisation of oversampling approaches has a beneficial impact on improving the classifier's performance. It is noteworthy that the Nearest Neighbour algorithm attained a precision of 100% when applied to the original dataset. However, there is a decrease observed in performance when using alternative approaches, with B-SMOTE exhibiting the highest similarity to the original performance with a value of 0.997603. This implies that although the Nearest Neighbour algorithm can accurately classify positive instances in some situations, its performance may vary when applied to datasets with diverse distributions. The MLP algorithm demonstrated a noteworthy level of accuracy when combined with K-CGAN, SMOTE, ADASYN, and B-SMOTE, all achieving a score close to 0.998. The oversampling approaches implemented in this study have resulted in a notable and statistically significant improvement in performance, as evidenced by the original dataset's precision of 0.990099. Finally, the precision of Logistic Regression reaches its maximum value when combined with K-CGAN (0.991221) and B-SMOTE (0.994725). Nevertheless, the performance of the model while utilising ADASYN was significantly worse, with an accuracy score of 0.909084. The aforementioned disparity underscores the susceptibility of Logistic Regression to the specific oversampling methodology utilised. In summary, it can be observed that various oversampling strategies elicit distinct responses from different classifiers. However, it is apparent that techniques such as B-SMOTE, SMOTE, K-CGAN, and ADASYN consistently improve precision across many classifiers in comparison to the original dataset. The selection of an oversampling strategy can have a substantial impact on the efficacy of a classifier, particularly in situations when precision is of utmost significance.

Table 42: Precision values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.982405	0.988636	0.980831	0.986072	0.981818	0.997998	0.999467	0.999182	0.999816	0.997085
Random Forest	0.982249	0.980170	0.977564	0.986111	0.981651	0.999598	0.999762	0.999760	0.999958	0.994135
Nearest Neighbour	0.961194	0.954416	0.954545	0.966197	1.000000	0.990056	0.982366	0.973762	0.997603	0.960606
MLP	0.959885	0.974504	0.962145	0.957219	0.990099	0.998400	0.997690	0.997970	0.998082	0.982456
Logistic Regression	0.968051	0.958457	0.949495	0.970149	0.989583	0.991221	0.974443	0.909084	0.994725	0.965732

The Table 43, which presented the recall values of various categorization systems using different oversampling procedures, presents noteworthy patterns of performance for different classifiers. The K-CGAN, SMOTE, ADASYN, and B-SMOTE resulted in notable enhancements in the recall performance of XGBoost. All of these strategies successfully attained or roughly approximated the maximum recall value of 1. The oversampling approaches implemented in this study demonstrated significant improvements in comparison to the original dataset, which had an XGBoost recall rate of 0.827068. The performance of Random Forest shown similarities to that of XGBoost in various aspects. The recall value obtained from the original dataset was 0.812030. Nevertheless, the utilisation of K-CGAN, SMOTE, and ADASYN resulted in a significant increase in recall performance, approaching a state of perfection. This outcome underscores the crucial contribution of these oversampling techniques in enhancing the efficacy of fraud detection by accurately identifying fraudulent transactions. The recall of the Nearest Neighbour algorithm exhibited a noticeable decrease when applied to the original dataset, yielding a value of 0.721804. Nevertheless, the implementation of algorithms such as K-CGAN, SMOTE, and ADASYN significantly enhanced the recall values, approaching a value of 1. The significant increase depicted in this scenario highlights the model's varying performance across diverse data distributions and emphasises the importance of employing selective data augmentation techniques. The MLP model achieved a recall of 0.842105 when evaluated using the

original dataset. However, the performance of the MLP model was significantly surpassed by the results obtained from employing K-CGAN, SMOTE, ADASYN, and B-SMOTE techniques. Once more, the utilisation of oversampling approaches has further emphasised the capability of the MLP in accurately classifying the majority of fraudulent transactions. The findings of the Logistic Regression analysis revealed a somewhat distinct narrative. The classifier's recall rate, when evaluated using the original dataset, was found to be 0.609023, which was the lowest among all the classifiers. Although the memory rate significantly improved to 0.999608 when K-CGAN was used in combination, the recall rates achieved with SMOTE and ADASYN were comparatively lower. The B-SMOTE approach demonstrated a significant enhancement in performance, resulting in a recall score of 0.996383. In conclusion, the recall values shown a general increase when utilising oversampling approaches such as K-CGAN, SMOTE, ADASYN and B-SMOTE in conjunction with various classifiers. This improvement was particularly notable when compared to the performance achieved on the original dataset. This underscores the impact of these methods in enhancing the ability of classifiers to accurately identify positive cases, which is a critical factor in domains such as credit card fraud detection where the failure to recognise a positive example can result in serious consequences.

Table 43: Recall values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.941011	0.932976	0.962382	0.917098	0.827068	0.999706	1.000000	0.999986	0.999703	0.955307
Random Forest	0.932584	0.927614	0.956113	0.919689	0.812030	0.999706	1.000000	1.000000	0.999661	0.946927
Nearest Neighbour	0.904494	0.898123	0.921630	0.888601	0.721804	0.999706	0.999804	1.000000	0.999746	0.885475
MLP	0.941011	0.922252	0.956113	0.927461	0.842105	0.999594	1.000000	0.999929	0.999746	0.938547
Logistic Regression	0.851124	0.865952	0.884013	0.841969	0.609023	0.999608	0.919681	0.860942	0.996383	0.865922

By doing an analysis of Table 44, which presents the F1 Score values for various oversampling approaches, valuable insights can be obtained regarding the performance of the classifiers. The application of oversampling techniques such as K-CGAN, SMOTE, ADASYN and B-SMOTE resulted in a significant improvement of XGBoost's F1 score, elevating it to the high 0.99 range. The improvements achieved by these strategies were clear when comparing their performance on the original dataset, where the F1 score was 0.873016. The F1 scores of Random Forest also experienced a significant improvement as a result of employing these oversampling techniques. The classifier demonstrated a satisfactory F1 score of 0.867470 when evaluated with the original dataset. However, when combined with advanced approaches like as K-CGAN, SMOTE, ADASYN and B-SMOTE, the classifier's performance significantly improved, approaching near-perfect results. The Nearest Neighbour algorithm demonstrated a relatively lower F1 score of 0.786885 on the original dataset. However, the utilisation of oversampling methods, specifically K-CGAN, SMOTE and B-SMOTE, substantially improved its performance, resulting in scores that approached the 0.99 range. The performance of MLP exhibited a remarkable level of consistency across several scenarios, demonstrating exceptional scores when combined with methodologies like as K-CGAN, SMOTE, ADASYN and B-SMOTE. The improved proficiency of the classifier is seen in the striking difference between the high scores achieved and the F1 score of 0.861538 obtained on the original dataset. Logistic regression shown considerable variability in its performance. The initial dataset exhibited an F1 score of 0.723214, however the use of the K-CGAN approach resulted in a substantial improvement, elevating the F1 score to 0.992409. Nevertheless, the utilisation of SMOTE and ADASYN resulted in a very restrained performance. Conversely, the implementation of B-SMOTE yielded a commendable score of 0.995553.

In summary, the findings shown in Table 44 underscore the significant impact of oversampling strategies on the F1 scores of classifiers, highlighting their transformative nature. Methods such as K-CGAN, SMOTE, ADASYN, and B-SMOTE have repeatedly facilitated classifiers in achieving a fair trade-off between precision and recall. This balance is crucial in fraud detection scenarios, where the occurrence of both false positives and FNs needs to be minimised.

Table 44: F1 Score values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.961263	0.960000	0.971519	0.950336	0.873016	0.997599	0.999733	0.999584	0.999760	0.975749
Random Forest	0.956772	0.953168	0.966720	0.951743	0.867470	0.997394	0.999881	0.999880	0.999809	0.969957
Nearest Neighbour	0.931983	0.925414	0.937799	0.925776	0.786885	0.992820	0.991008	0.986707	0.998673	0.921512
MLP	0.950355	0.947658	0.959119	0.942105	0.861538	0.998400	0.998844	0.998949	0.998913	0.960000
Logistic Regression	0.905830	0.909859	0.915584	0.901526	0.723214	0.992409	0.946270	0.884358	0.995553	0.913108

From Table 45 it is evident that the accuracy of XGBoost exhibited a remarkable level of consistency across all observed scenarios. The accuracy of nearly every approach remained consistently high, typically in the range of 0.999. The K-CGAN technique had a notable performance, achieving a score of 0.997599, which remained remarkably high. The performance trajectory of Random Forest exhibited similarities to that of XGBoost. The accuracy of the majority of approaches remained consistently excellent, hovering in the range of 0.999. The implementation of the K-CGAN technique resulted in a noteworthy value of 0.997394. In comparison, the performance of the Nearest Neighbour classifier exhibited greater variability. A considerable portion of the approaches exhibited a high level of accuracy, consistently achieving scores in the range of 0.999. However, when combined with the K-CGAN, SMOTE, and ADASYN methods, these methods had more noticeable decreases in accuracy, resulting in scores ranging from 0.992 to 0.986. MLP continued to demonstrate strong performance throughout the specified period. With the exception of the K-CGAN approach, which achieved an accuracy of 0.998400, the majority of the other strategies consistently maintained MLP's accuracy in the range of 0.999. While the observed discrepancy with K-CGAN was quite small, it was nonetheless perceptible. Logistic Regression exhibited the highest level of dynamism in terms of the variability of its accuracy across different approaches. Multiple strategies demonstrated the ability to maintain a performance level over 0.999. However, certain methods, such as K-CGAN and particularly ADASYN, exhibited slight declines in performance. K-CGAN achieving 0.992409. The ADASYN, specifically, reduced the accuracy to 0.887842. Remarkably, in the middle of these fluctuations, the VANILLA CGAN approach demonstrated a resurgence in its accuracy, reaching a robust value of 0.999174. In conclusion, the findings from Table 45 highlight the constant performance excellence of XGBoost and Random Forest algorithms when applied to various oversampling techniques. Although each classifier exhibited varying degrees of sensitivity towards specific methods, the general pattern seen suggests that these oversampling techniques have a positive impact on enhancing accuracy. Nevertheless, there were subtle distinctions, and specific pairings such as Nearest Neighbour with K-CGAN or Logistic Regression with ADASYN demonstrated that selecting the appropriate oversampling technique in conjunction with the classifier is crucial for achieving optimal results.

Table 45: Accuracy values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.999622	0.999594	0.999748	0.999482	0.999551	0.997599	0.999733	0.999585	0.999761	0.999762
Random Forest	0.999580	0.999524	0.999706	0.999496	0.999537	0.997394	0.999880	0.999880	0.999810	0.999706
Nearest Neighbour	0.999342	0.999244	0.999454	0.999230	0.999270	0.992820	0.990905	0.986578	0.998678	0.999244
MLP	0.999510	0.999468	0.999636	0.999384	0.999494	0.998400	0.998839	0.998952	0.998917	0.999608
Logistic Regression	0.999118	0.999104	0.999272	0.999006	0.999129	0.992409	0.947643	0.887842	0.995568	0.999174

4.6.6 Impact of Oversampling using Novelty K-CGAN

The utilisation of the novelty K-CGAN to oversample the minority class in credit card fraud detection dataset has demonstrated several significant impacts. Firstly, the application of the novelty K-CGAN led to a dataset with equal numbers of fraudulent and valid transactions, achieving a balanced class distribution. This balance resulted in an overall performance improvement in the model, enhancing the model's capability to identify fraudulent transactions. Moreover, the performance metrics of the models, including precision, recall, and F1 scores, remarkably improved following the oversampling process. These metrics indicated that the novelty of K-CGAN's application in addressing class imbalance resulted in a better credit card fraud detection performance. By generating synthetic fraud transactions that were representative of the actual fraudulent transactions, the novelty custom K-CGAN ensured an effective augmentation of the dataset. This further boosted the classifiers' overall performance.

Notably, the oversampling approach using the novelty K-CGAN also helped to alleviate the class imbalance issue, which is a common challenge in credit card fraud detection datasets. By creating synthetic fraud transactions that closely resembled real fraudulent activities, the models became more adept at identifying and classifying fraudulent cases accurately. Furthermore, the models generated high precision and recall values, which are critical metrics for fraud detection. The ability to minimise false positives while identifying as many fraud cases as possible is crucial in maintaining the effectiveness of fraud detection systems. The novelty K-CGAN's oversampling approach delivered impressive results in this regard, improving the precision and recall rates significantly.

In summary, the utilisation of the novelty K-CGAN in oversampling the minority class in credit card fraud detection datasets has shown remarkable benefits. By addressing class imbalance, generating representative synthetic fraud transactions, and improving precision and recall rates, the novelty K-CGAN contributes to the enhanced credit card fraud detection performance and the overall reliability of the models.

4.7 Optimized Novelty Loss Evaluation comparison breast cancer data

The entire original dataset was used for training and 80-20% split was not necessary since we planned to conduct testing with the synthetic dataset. During training using the novelty GAN, we generated 200 synthetic samples of both benign and malignant tumours after a certain number of epochs. We then used this data to calculate the quality score. Once training is completed, these 200 benign and 200 malignant synthetic samples are generated using the Novelty loss K-CGAN and this data is used to check the quality of synthetic dataset.

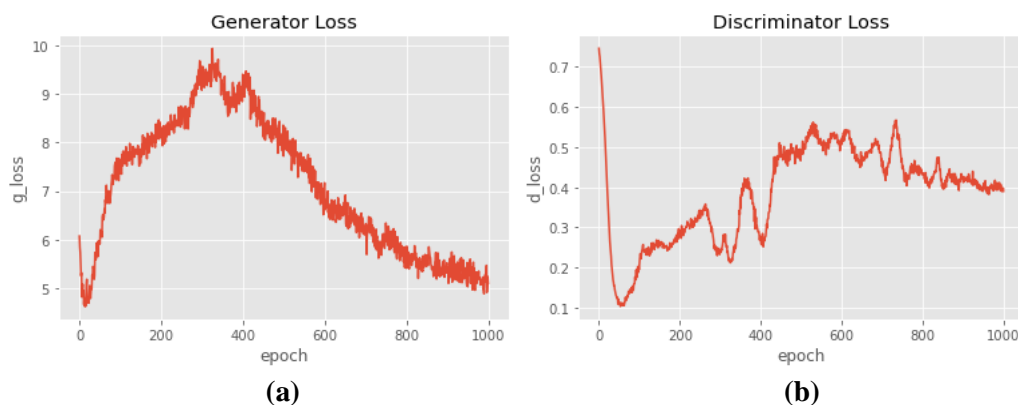


Figure 23: K-CGAN generator (a) and discriminator (b) loss breast cancer data

Figure 23 (a) shows the generator of K-CGAN on breast cancer data. The generator loss initially increased up to 300 epochs, but then decreased smoothly and stabilised around 4.0. On the other hand, Figure 23 (b) presents the discriminator of K-CGAN. The discriminator loss initially increased up to 300 epochs, then stabilised between 0.4 to 0.5 for the next 700 epochs. These results as shown in Figure 23 indicate that the K-CGAN architecture was able to generate synthetic samples that are similar enough to the real samples, as evidenced by the stable generator loss and the discriminator's inability to distinguish between real and synthetic samples. The stability of the loss values after a certain number of epochs indicates that the GAN has converged to a stable state, where further training would not significantly improve the performance of the model. The results concludes that the generator loss first rose up to 300 epochs before gradually declining and stabilising at around 4.0. After initially rising up to 300 epochs, the discriminator loss stabilised at 0.4 and 0.5 over the following 700 epochs. The steady generator loss and the discriminator's inability to discriminate between actual and synthetic samples show that the K-CGAN architecture was able to produce synthetic samples that are sufficiently comparable to the real examples. Further, the continuous generator loss and the discriminator's inability to distinguish between real and artificial samples demonstrate that the K-CGAN architecture was successful in creating artificial samples that are sufficiently similar to the real instances (Ding et al., 2023). After a given number of epochs, the loss values become stable, indicating that the K-CGAN has reached a stable state where further training will not appreciably enhance the model's performance (Li et al., 2020).

Balanced Dataset

In order to address the class imbalance issue in the dataset, we implemented oversampling techniques to increase the number of samples for the minority class of malignant cases from 212 to 357, as shown in Table 46. In the series of well-designed experiments that followed, we used advanced GAN-based models that

were carefully optimised and trained. We also employed various oversampling techniques such as ADASYN, B-SMOTE and SMOTE to augment the dataset. Our main goal was to thoroughly evaluate and compare the performance and effectiveness of each model. Additionally, we conducted a meticulous analysis of the classification performance of these models and compared the data quality with the original dataset. Through these rigorous experiments, we aimed to gain deeper insights into the capabilities and limitations of these techniques. This allowed us to comprehensively examine and compare the classification performance of each model and the impact of addressing imbalanced class representation in classification tasks.

Table 46: Balanced breast cancer dataset using optimised methods

Description	Value
Benign	357
Malignant	357

4.7.1 Hyperparameter Settings

Oversampling Methods

Table 47: Oversampling methods hyperparameter settings

Method	Settings
SMOTE	default number of nearest neighbors is 5 (imbalanced-learn.org, n.d.)
ADASYN	default number of nearest neighbors is 5 and the default ‘synthetic’ points per minority class sample is set to 10 (imbalanced-learn.org, n.d.)
B-SMOTE	default number of nearest neighbors for B-SMOTE is 5, the default ‘synthetic’ points per minority class sample is set to 10, and the maximum number of synthetic points that can be generated is 20 (imbalanced-learn.org, n.d.)

As indicated by Table 47, the default number of nearest neighbors for SMOTE is 5, as specified on imbalanced-learn.org. This means that, for each sample in the minority class, SMOTE selects 5 closest neighbors and generates synthetic samples along the line segments connecting the sample to its neighbors. The aim is to enhance the representation of the minority class. ADASYN also employs the default number of nearest neighbors, which is 5. Additionally, ADASYN introduces the concept of density distribution. It creates extra synthetic samples for minority class samples that exist in densely populated regions, effectively addressing the issue of imbalance. The default number of ‘synthetic’ points per minority class sample is set to 10. Consequently, ADASYN produces 10 synthetic samples for each minority class sample, further reinforcing the representation of the minority class. Similar to ADASYN, B-SMOTE generates synthetic samples. The default ‘synthetic’ points per minority class sample is configured to 10, with a maximum of 20 synthetic points that can be generated, ensuring controlled oversampling.

Table 48: Vanilla CGAN, WGAN and NS GAN optimised hyperparameter settings

Hyperparameter	Vanilla cGAN	WGAN	NS GAN
Activation	Leaky ReLU	Relu	LeakyReLU
Batch Size	32	16	64
Dropout	0.5	0.1	0.5
Optimizer	RMSProp	AdaGRAD	Adam
Learning Rate	0.0001	0.001	0.001
Discriminator Layers	128,64,32	128,64,32	128,64,32
Generator Layers	64,32	64,32	64,32

The hyperparameters for the Vanilla CGAN are thoughtfully selected to optimise its performance in generating breast cancer data samples are listed in Table 48. The Leaky ReLU activation function is used to introduce non-linearity and address vanishing gradient problems. A batch size of 32 strikes a balance between computational efficiency and gradient accuracy during training. To prevent overfitting and improve the model's generalisation capabilities, a dropout rate of 0.5 is implemented. The RMSProp optimizer is chosen to optimise the model's parameters, with a learning rate of 0.0001 ensuring controlled and steady convergence. The model's architecture includes three discriminator layers with 128, 64, and 32 nodes, respectively, enabling it to identify complex data patterns. On the generator side, there are two layers with 64 and 32 nodes, respectively, facilitating the capture of the underlying data distribution.

The hyperparameters for the WGAN are carefully chosen to optimise its performance in generating breast cancer data are also shown in Table 48. The Relu activation function is selected for its ability to handle intricate non-linear relationships within the data. By using a batch size of 16, a balance is achieved between computational efficiency and gradient stability during training. To prevent overfitting and improve generalisation, a dropout rate of 0.1 is applied. The AdaGRAD optimizer is utilised to fine-tune the model's weights, enhancing convergence speed and accuracy. With a learning rate of 0.001, the model ensures a controlled and steady learning process. The architecture consists of three discriminator layers with 128, 64, and 32 nodes, allowing it to discern complex patterns in the data. Meanwhile, the generator network incorporates two layers with 64 and 32 nodes respectively, effectively capturing the underlying data distribution.

The NS GAN utilises carefully selected hyperparameters to optimise breast cancer data sample generation are demonstrated in Table 48. The LeakyReLU activation function is chosen for introducing non-linearity and addressing the vanishing gradient problem. A batch size of 64 balances computational efficiency and gradient accuracy. A dropout rate of 0.5 prevents overfitting and improves generalisation. The Adam optimizer with a learning rate of 0.001 ensures controlled and gradual learning. The model's architecture consists of three discriminator layers (128, 64, and 32 nodes) and two generator layers (64 and 32 nodes) to discern intricate patterns and capture data distribution effectively.

Table 49: LS GAN and SDG GAN optimised hyperparameter settings

Hyperparameter	LS GAN	SDG GAN
Activation	LeakyReLU	LeakyReLU
Batch Size	32	64
Dropout	0.5	0.1
Optimizer	RMSProp	RMSProp
Learning Rate	0.0001	0.001
Discriminator Layers	128,64,32	256,128,64
Generator Layers	64,32	64,32

Table 49 presents the hyperparameter settings for the LS GAN and SDG GAN models. These models utilise these optimised hyperparameter configuration parameters to achieve optimal performance for breast cancer data. The table provides detailed information on the hyperparameters for the activation function, batch size, dropout rate, optimizer, learning rate, discriminator layers, and generator layers. For the LS GAN model, the activation function used is LeakyReLU, which introduces a small amount of negative slope to prevent dead neurons. The batch size is set to 32, ensuring efficient processing of training samples in each iteration. The dropout rate is set to 0.5, aiding in regularisation and reducing overfitting. The optimizer used is RMSProp, which adapts the learning rate based on the magnitude of gradient updates. The learning rate is set to 0.0001, controlling the step size during training. The discriminator in the LS GAN model consists of three hidden layers with 128, 64 and 32 neurons, respectively. These layers capture complex patterns and features in the input data. On the other hand, the generator comprises two hidden layers with 64 and 32 neurons, generating synthetic samples that resemble the real data distribution. Moving on to the SDG GAN model, it also utilises LeakyReLU as the activation function to introduce non-linearity. The batch size is set to 64, allowing for larger mini-batch training. The dropout rate is set to 0.1, providing regularisation to prevent overfitting. Similar to LS GAN, RMSProp is used as the optimizer, and the learning rate is set to 0.001. The discriminator in the SDG GAN model is composed of three hidden layers with 256, 128, and 64 neurons, respectively, enabling the model to learn high-level representations. The generator of SDG GAN, like LS GAN, has two hidden layers with 64 and 32 neurons, generating diverse and realistic synthetic samples. By fine-tuning these hyperparameters, we're able to achieve optimised settings for both LS GAN and SDG GAN models.

Table 50: Novelty K-CGAN optimised hyperparameter settings for breast cancer data

Hyperparameter	Generator Neural Network	Discriminator Neural Network
Activation	ReLU	LeakyReLU
Loss function	Modified Binary Cross Entropy + KL Divergence	Binary Cross Entropy
Hidden Layers	(3 - 2 hidden, 1 output) 64, 32, 29	(3 - 2 hidden, 1 output) 20, 15, 1
Dropout	0.2	0.2
Output Optimizer	Adam	Adam
Learning Rate	0.0001	0.0001
Random Noise Vector	100	-
Kernel Initializer	glorot_uniform	-
Kernel Regularizer	L2 method	L2 method
Total Learning Parameters	10,226	1,386

Table 50 presents the custom hyperparameter settings for the K-CGAN model applied to the breast cancer dataset. These are specific optimised hyperparameters for both the generator and discriminator neural networks. The generator neural network comprises two hidden layers, with 64 and 32 neurons respectively, ensuring a robust representation of the data. The activation function employed in the generator neural network is ReLU, known for its effectiveness in capturing complex patterns. To optimise the generator's performance, the loss function combines the trained discriminator loss and Kullback-Leibler (KL) divergence, promoting both accuracy and diversity in generated samples. The output optimizer utilised is Adam, a popular choice for its adaptive learning rate capabilities, and the learning rate is set to 0.0001. Additionally, a dropout layer with a dropout rate of 0.2 is included to prevent overfitting and enhance generalisation. The generation process is injected with variability by the random noise vector, which has a dimension of 100. Convergence and stability are promoted by applying the glorot_uniform kernel initializer. To mitigate overfitting, a kernel regularizer based on the L2 method is employed. The complexity and capacity of the model are reflected in its total of 10,226 learning parameters.

Further, the discriminator neural network also consists of two hidden layers, with 20 and 15 neurons respectively, ensuring discriminative power. The activation function used in the discriminator neural network is LeakyReLU, which allows the model to capture negative evidence and improve performance. The loss function employed is binary cross-entropy, enabling effective discrimination between real and generated samples. Similar to the generator, the output optimizer is Adam with a learning rate of 0.0001. To further improve robustness, a dropout layer with a dropout rate of 0.2 is integrated into the discriminator neural network. The L2 method is utilized as a kernel regularizer to prevent overfitting. With a total of 1,386 learning parameters, the model showcases a relatively compact design. These detailed custom hyperparameter settings provide a solid foundation for the K-CGAN model, ensuring its efficacy and performance in generating realistic and high-quality samples for breast cancer data.

Classification Methods

In our methodology as shown in Table 51, we utilised the Random Forest algorithm due to its robustness and versatility in handling complex datasets. To ensure controlled randomness during the sampling process, we specifically set the `random_state` parameter to 42. By maintaining this default value, we intentionally guarantee consistent organisation of the dataset during random sampling. As a result, the same training and testing subsets are consistently derived, promoting uniformity in subsequent analyses and evaluations. This strategic choice aligns with established best practices and enhances the reliability of our findings (scikit-learn.org, n.d.). To leverage the effectiveness of ensemble learning, we turned to XGBoost for its demonstrated prowess in predictive modelling. In line with our commitment to reproducibility and controlled randomness, we set the `random_state` parameter to 42. This ensures that the random aspects of XGBoost, such as sample selection and splitting, follow a consistent pattern across different runs. This alignment with our overarching goal of maintaining consistency and comparability in our experimentation and analysis strengthens the reliability of our results (scikit-learn.org, n.d.). To harness the power of neighbor-based predictions for proximity-based classification tasks, we employed the KNN algorithm. In our pursuit of optimal performance, we fine-tuned the hyperparameter `n_neighbors` and set it to 100. This choice dictates that each prediction is influenced by the 100 nearest neighbors in the dataset, emphasising local patterns and relationships. This strategic decision enhances the algorithm's ability to capture intricate patterns within the data, contributing to the reliability of our classification outcomes (scikit-learn.org, n.d.). Our exploration of deep learning methodologies led us to adopt the MLP classification method, renowned for its capacity to handle complex data representations. To ensure efficient and effective model training, we configured the `max_iter` parameter to represent the maximum number of training epochs. This designation accounts for the possibility that the learning process might converge before reaching the specified maximum. In our experiment, we set this value to 300, striking a balance between comprehensive training and computational efficiency. Additionally, to exert control over randomness and enable reproducibility, we assigned the `random_state` a value of 1. This strategic choice aligns with best practices, facilitating the consistent replication of our results across different iterations (scikit-learn.org, n.d.).

Table 51: Classification methods hyperparameter configuration settings

Method	Settings
Random Forest	<p>To control randomness of the sample <code>random_state</code> was set to 42, by setting the default value we're ensuring that the data is getting arranged the same way, as a result it returns the same training and testing subsets.</p> <p>(scikit-learn.org, n.d.)</p>
XGBoost	<p>To control randomness of the sample <code>random_state</code> was set to 42</p> <p>(scikit-learn.org, n.d.)</p>
KNearest Neighbor	<p>The tuning hyper parameter <code>n_neighbors</code> was set to 100</p> <p>(scikit-learn.org, n.d.)</p>
MLP	<p>The <code>max_iter</code> parameter represents the maximum number of epochs for model training. It is referred to as "maximum" because the learning process may stop before reaching the maximum number of iterations, depending on other termination criteria, we have set it to 300.</p> <p>To control the random factor (<code>random_state</code>) was set to 1. It's recommended to set the seed for the random generator to confirm that the outcomes can be consistently reproduced.</p> <p><code>random_state=1, max_iter=300</code></p> <p>(scikit-learn.org, n.d.)</p>

4.7.2 Results Analysis

Distribution Charts

Correlation

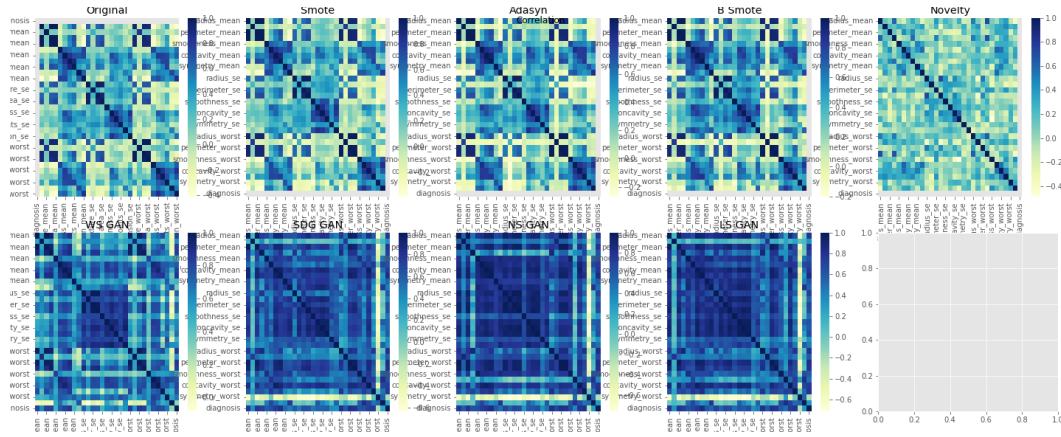
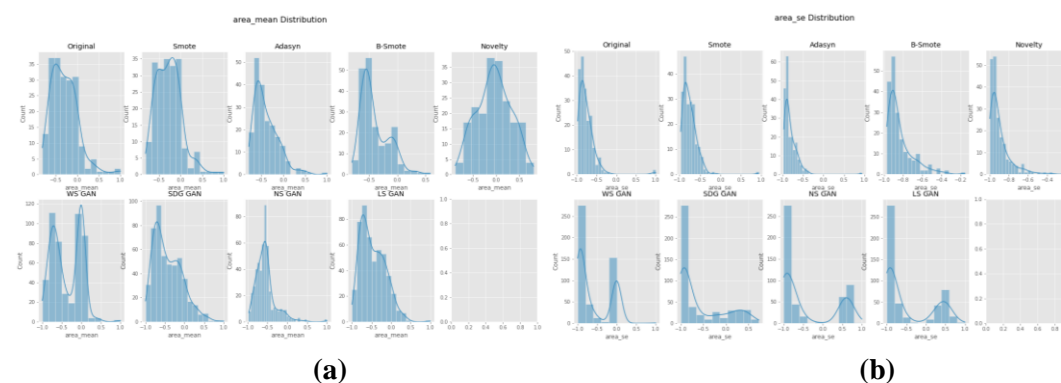


Figure 24: Correlation metric comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN breast cancer data

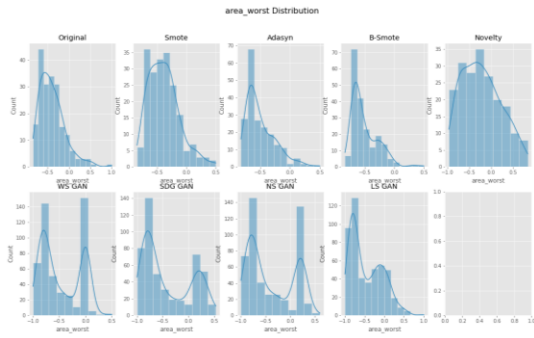
Figure 24 displays the correlation metric comparison of the original dataset with multiple methods. The findings effectively demonstrate that K-CGAN excels in preserving the original structure of the breast cancer dataset. Notably, K-CGAN outperforms other GAN-based methods by avoiding the introduction of bias or noise into the data. This indicates the suitability of our method for this specific dataset and further strengthens its effectiveness in maintaining the original structure. Additionally, the heatmap visually showcases the correlation patterns among various features, providing valuable insights into the relationship between multiple variables. The darker colours, closer to 1 or -1, indicate stronger correlations, either positive or negative, while the lighter colours, closer to 0, indicate weaker correlations or no correlation at all. This detailed analysis enhances our understanding of the dataset and its underlying characteristics.

Single Column Distribution

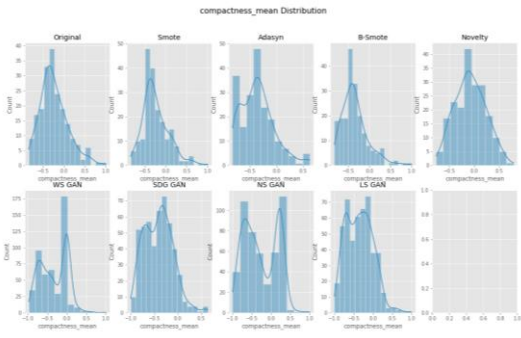


(a)

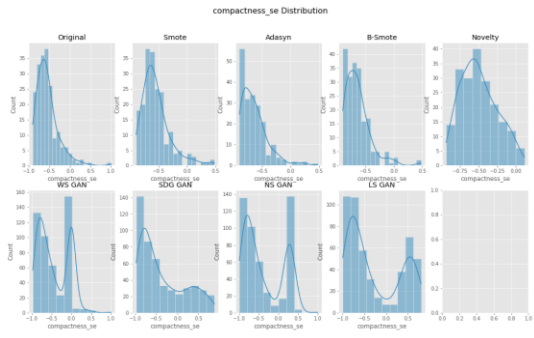
(b)



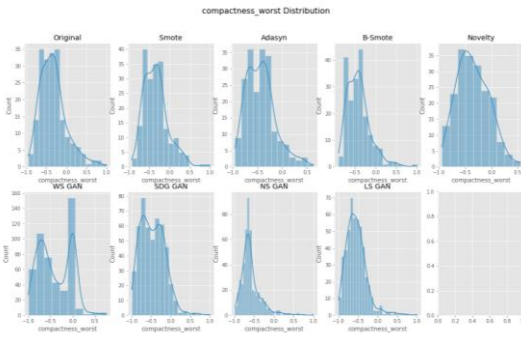
(c)



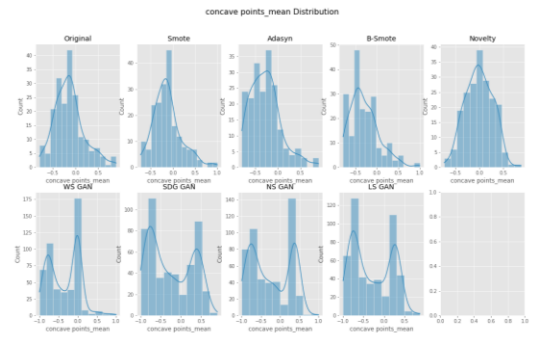
(d)



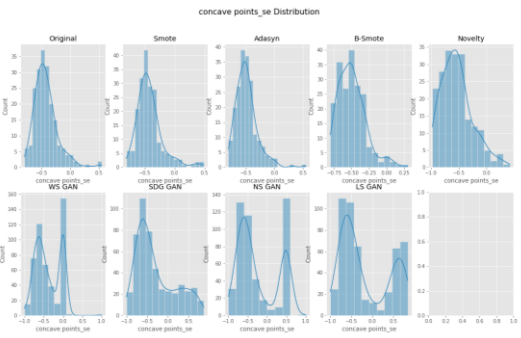
(e)



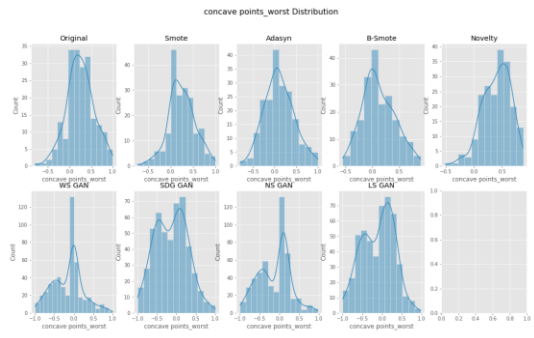
(f)



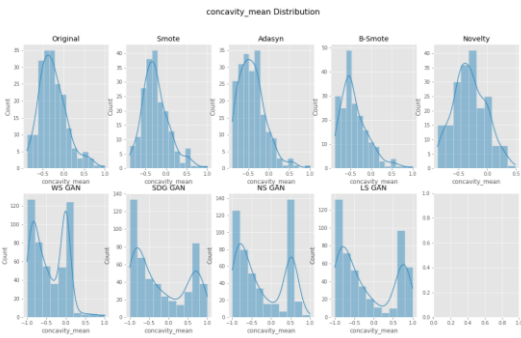
(g)



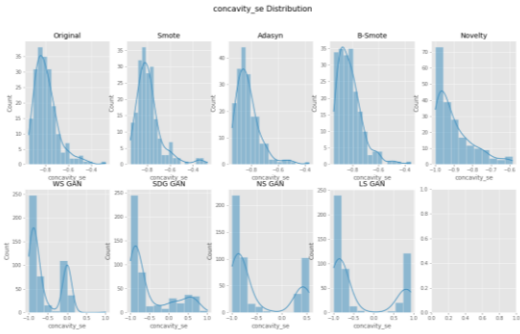
(h)



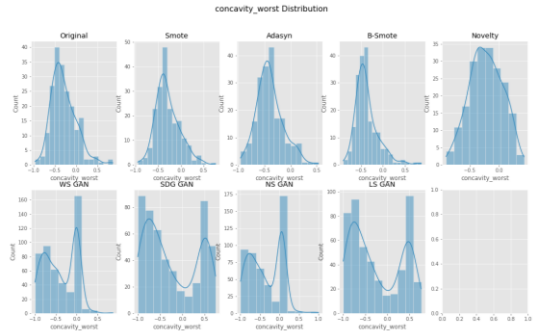
(i)



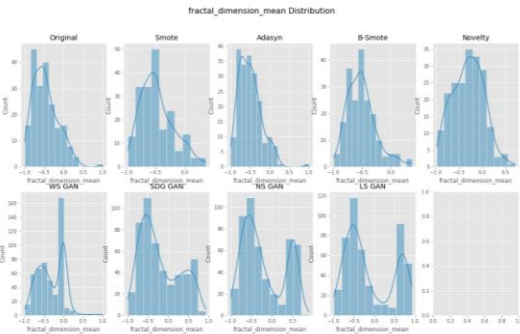
(j)



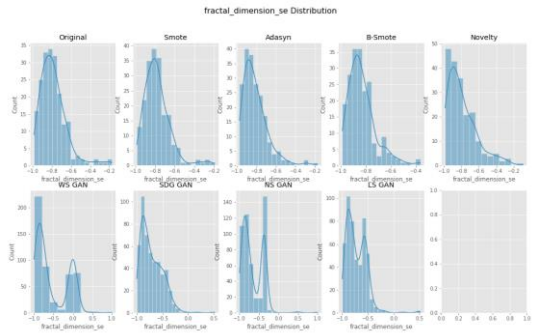
(k)



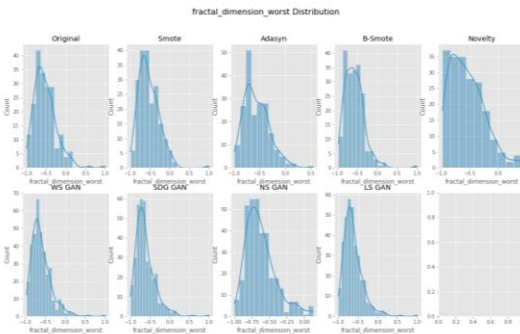
(l)



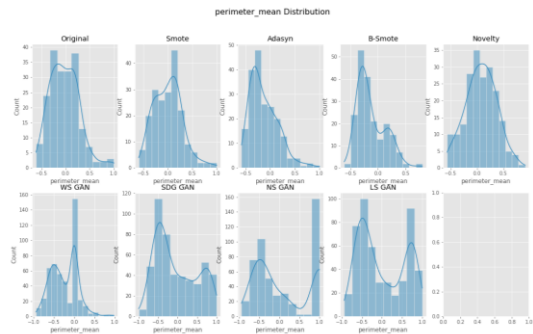
(m)



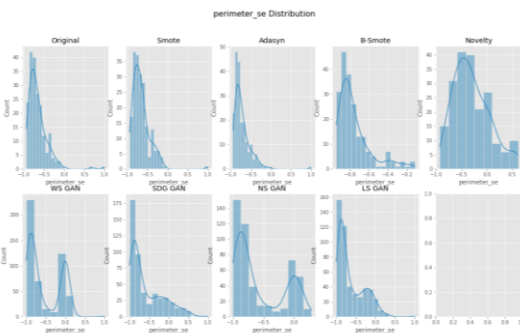
(n)



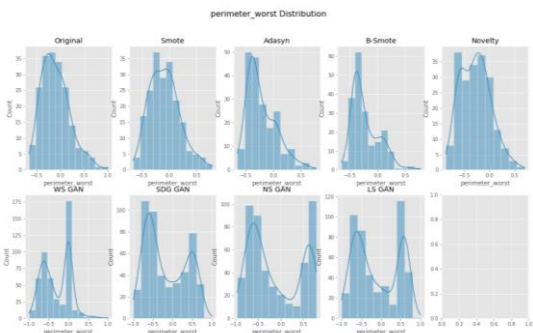
(o)



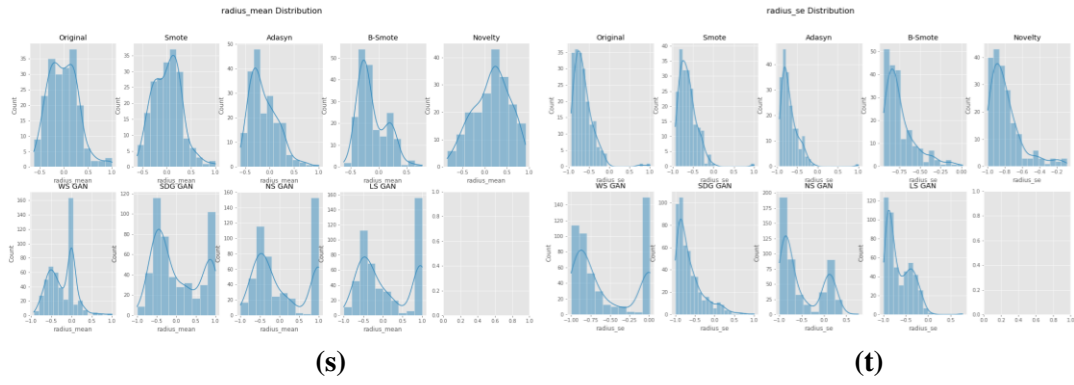
(p)



(q)

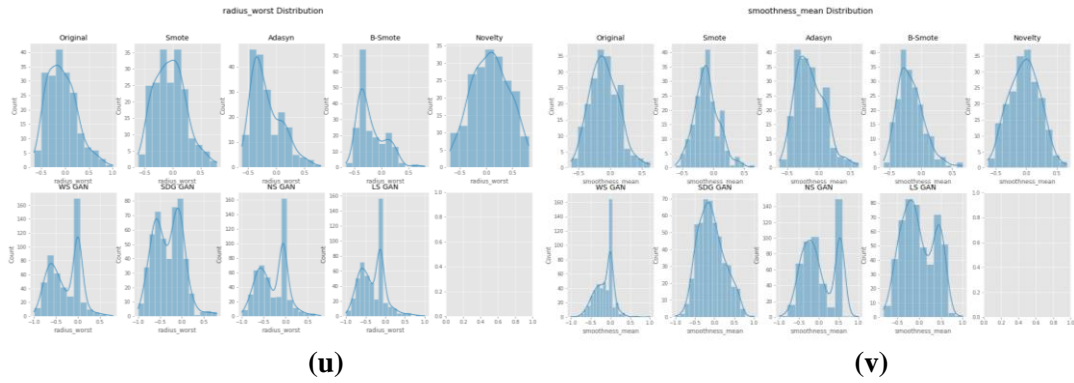


(r)



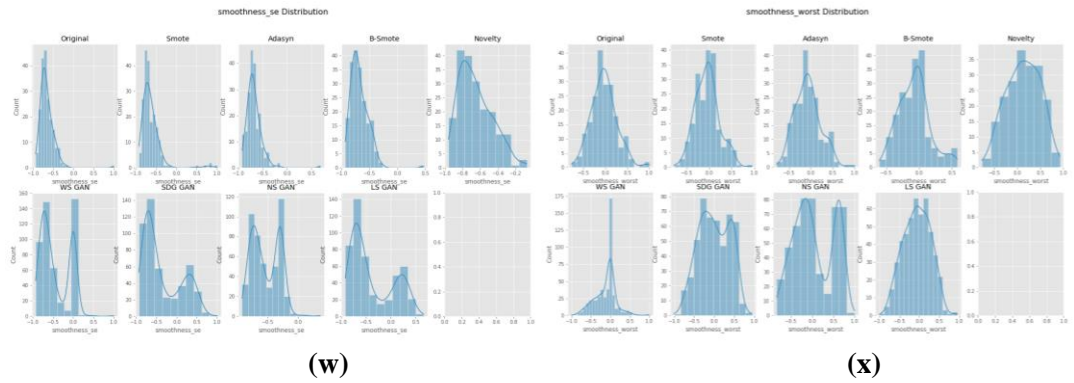
(s)

(t)



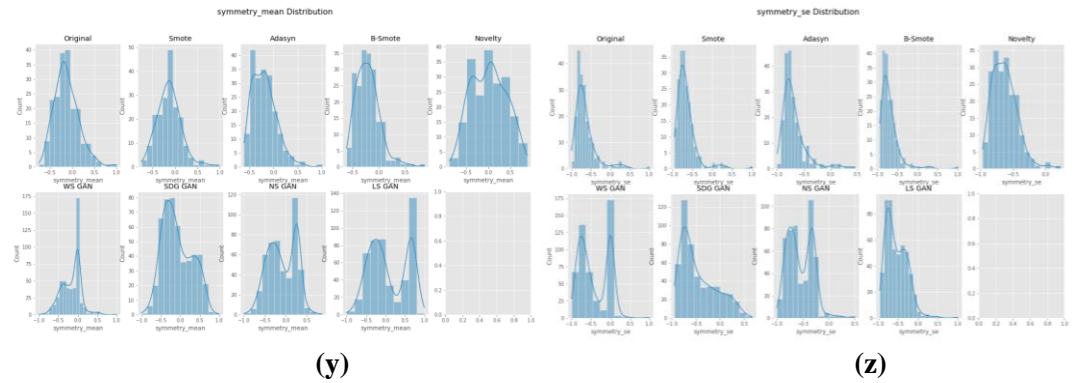
(u)

(v)



(w)

(x)



(y)

(z)

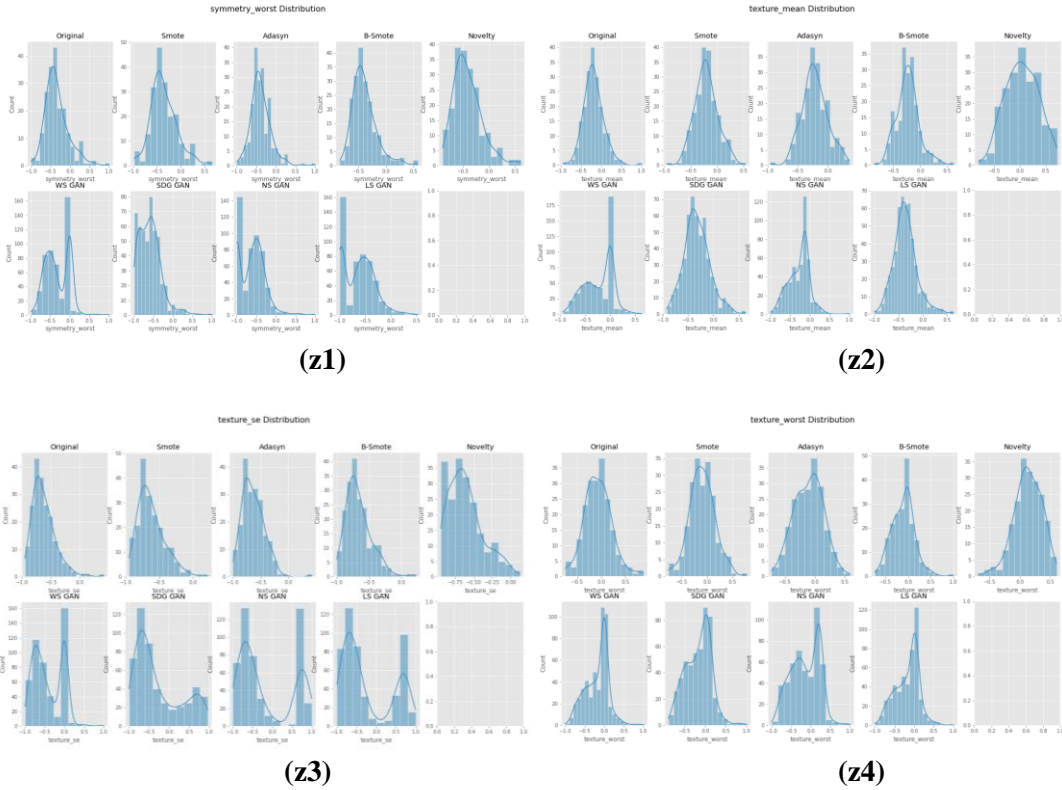


Figure 25: Single column ‘area_mean’ distribution comparison of Original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN breast cancer data (a), ‘area_se’ (b), ‘area_worst’ (c), ‘compactness_mean’ (d), ‘compactness_se’ (e), ‘compactness_worst’ (f), ‘concave points_mean’ (g), ‘concave points_se’ (h), ‘concave points_worst’ (i), ‘concavity_mean’ (j), ‘concavity_se’ (k), ‘concavity_worst’ (l), ‘fractal_dimension_mean’ (m), ‘fractal_dimension_se’ (n), fractal_dimension_worst’ (o), ‘perimeter_mean’ (p), ‘perimeter_se’ (q), ‘perimeter_worst’ (r), ‘radius_mean’ (s), ‘radius_se’ (t), ‘radius_worst’ (u), ‘smoothness_mean’ (v), ‘smoothness_se’ (w), ‘smoothness_worst’ (x), ‘symmetry_mean’ (y), ‘symmetry_se’ (z), ‘symmetry_worst’ (z1), ‘texture_mean’ (z2), ‘texture_se’ (z3), ‘texture_worst’ (z4)

Algorithm effectiveness can be evaluated by comparing the distribution of every feature across the dataset. Figure 25 (a) and (b) shows the single-columns ‘area_mean’ and ‘area_se’ distribution comparison between original, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with breast cancer dataset reveals that the methodologies employed have had a significant impact on the dataset. Moreover, it is evident that the K-CGAN achieved commendable quality. It can be seen that overall K-CGAN closely resembles the original data and its distance between the actual and generated samples is much lower than that of other GAN algorithms used in this study. This demonstrates a great potential for reducing data imbalance in breast cancer datasets and achieving an equitable distribution of values.

Figure 25 (c) and (d) illustrates the comparison of the single column ‘area_worst’ and ‘compactness_mean’ distribution among the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with breast cancer data. It is evident that while some algorithms were able to capture the true distribution of the data, others produced results with significant deviations. Overall, the Novelty K-CGAN approach emerged to be effective, accurately reflecting the quality of original dataset. The findings suggest that generative modelling of K-CGAN is a viable method for synthesising new data points for breast cancer data. Further comparisons, Figure 25 (e) and (f) illustrates the single columns ‘compactness_se’ and ‘compactness_worst’ distribution between the original dataset and algorithms such as

SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN, and K-CGAN with breast cancer data. Out of all the GAN-based algorithms, it was found that despite some deviations the K-CGAN improvement in predictability when using this algorithm over the others.

Figure 25 (g) and (h) displays the single column 'concave points_mean' and 'concave points_se' distribution comparison of various algorithms applied to breast cancer dataset. The results reveal that overall novelty K-CGAN achieved reasonable degree of similarity. Figure 25 (i) and (j) presents a single column 'concave points_worst' and 'concavity_mean' distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with breast cancer data. The results show that the novelty loss K-CGAN algorithm performed the best among all the GAN-based variants tested. This finding confirms that the K-CGAN approach is a viable solution for dealing with imbalanced datasets. It is also worth noting that among the other imbalanced learning algorithms, SMOTE and ADASYN achieved superior results.

Figure 25 (k) and (l) demonstrates the comparison of the single-column 'concavity_se' and 'concavity_worst' distribution between the original dataset and the synthetic datasets generated by SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with breast cancer data. Of all the GAN-based algorithms, overall K-CGAN showed the highest effectiveness in terms of data distribution. This performance is reflective of the capabilities of this algorithm and its potential to be used for other datasets. The results suggest that K-CGAN has strong capacity to generate synthetic datasets with closely resembling distributions as compared to the original dataset. Figure 25 (m) and (n) presents a comparison of the single column 'fractal_dimension_mean' and 'fractal_dimension_se' distribution in the original dataset alongside various synthetically generated by SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN. The overall output from the novel K-CGAN algorithm demonstrating superior performance in comparison to all other GAN-based algorithms. This result is indicative of the effectiveness of K-CGAN in producing accurate synthetic datasets from existing data sources. As such, K-CGAN can be considered a viable solution for generating synthetic datasets as part of predictive analytics projects.

Figure 25 (z) and (p) clearly shows the single column 'fractal_dimension_worst' and 'perimeter_mean' distribution comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN. Of all the algorithms used for balancing the breast cancer data set, K-CGAN demonstrated good standing results. Figure 25 (q) and (r) presents a comparison of the single column 'perimeter_se' and 'perimeter_worst' distribution between the original dataset and various data augmentation techniques SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN. It is evident that despite some deviations the K-CGAN model is stable in good standing. This result suggests that GANs, when used in conjunction with a novelty loss, can be an effective tool for data augmentation and improving classifier accuracy.

Figure 25 (s) and (t) illustrates a comparison of the single column 'radius_mean' and 'radius_se' distributions from the original dataset, as well as SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN. It is clear from the results that overall despite some deviations the K-CGAN algorithm produced the quality representation of the underlying data, outperforming all other GAN-based algorithms.

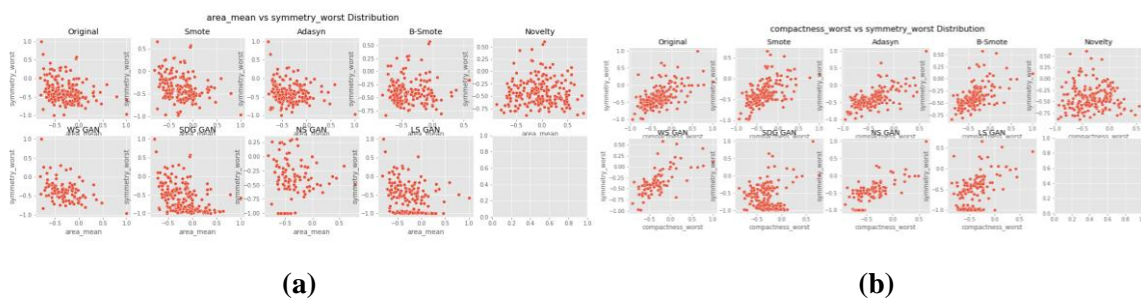
Figure 25 (u) and (v) demonstrates the single column comparison of 'radius_se' and 'radius_worst' distribution with the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN on breast cancer data. The results of the comparison indicate that K-CGAN outclasses all other GAN-based algorithms in terms of producing an quality representation of the underlying dataset. The univariate distribution in Figure 25 (v) shows that of all algorithms, K-CGAN has the closest resemblance to the original data 'smoothness_mean'. The distance between the actual and generated samples

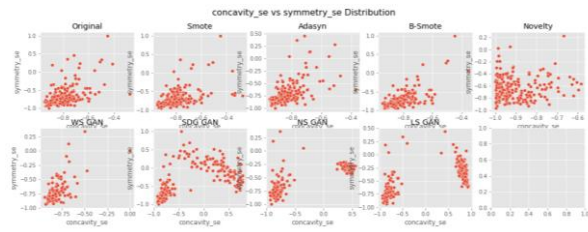
is significantly less when using K-CGAN compared with other GAN-based algorithms. This suggests that K-CGAN is more effective at producing synthetic data that resembles the original dataset than the other GAN-based algorithms. This indicates that K-CGAN is a viable solution for generating high quality, realistic datasets for use in machine learning applications.

Figure 25 (w) and (x) show that K-CGAN is effective algorithm for generating samples closely resembling those in the original dataset. The univariate comparison of 'smoothness_se' and 'smoothness_worst' distributions demonstrates this clearly; the distance between the actual and generated samples is far less in the case of K-CGAN compared to other GAN-based algorithms. This indicates that K-CGAN is efficient algorithm for generating samples that closely represent those in the original dataset. Figure 25 (y) and (z) illustrates the effectiveness of K-CGAN compared to other GAN-based algorithms. The single column comparison of 'symmetry_mean' and 'symmetry_se' distribution for the original dataset along with SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN reveals that K-CGAN overall resembles the original data. Furthermore, the distance between actual and generated samples overall is far lesser for K-CGAN than other methods. This emphasizes its effectiveness in generating synthetic data that mimics real-world characteristics. As a result, it can be safely assumed that K-CGAN is a promising algorithm for creating synthetic datasets which approximate real-world data.

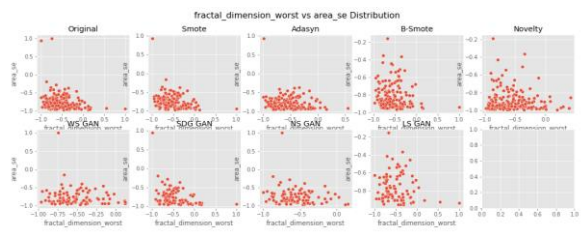
The univariate distribution of 'symmetry_worst' and 'texture_mean' in Figure 25 (z1) and (z2) provides a clear picture of the effectiveness of various algorithms on breast cancer data. The original data has the closest resemblance to K-CGAN, with a smaller distance between actual and generated samples than other GAN-based approaches. This indicates that K-CGAN is efficient at learning the characteristics of the original dataset and accurately reproducing them in generated data. Furthermore, K-CGAN also provides improved stability and controllability during training, resulting in better-quality synthetic data. Consequently, K-CGAN is a reliable method for generating high-fidelity synthetic datasets that closely represent real-world data distributions. Figure 25 (z3) and (z4) demonstrates the effectiveness of various algorithms in generating dataset samples that closely resemble the original data. The single column comparison presents the distribution of 'texture_se' and 'texture_worst' for original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN on breast cancer data. The univariate distribution indicates that the K-CGAN algorithm overall has the close resemblance to the original data. Furthermore, this is evidenced by the fact that distance between actual and generated samples is lesser in case of K-CGAN as compared with other GAN-based algorithms.

Bi-Variate Distribution

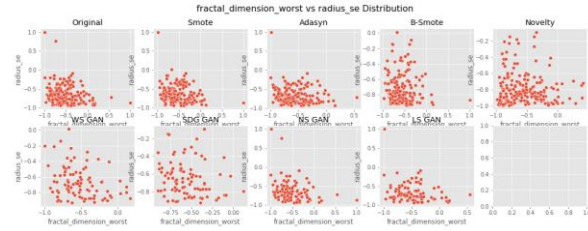




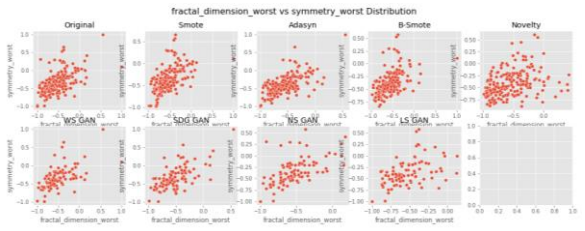
(c)



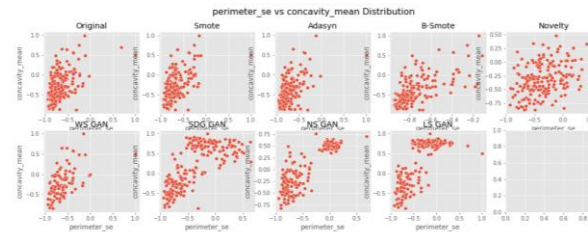
(d)



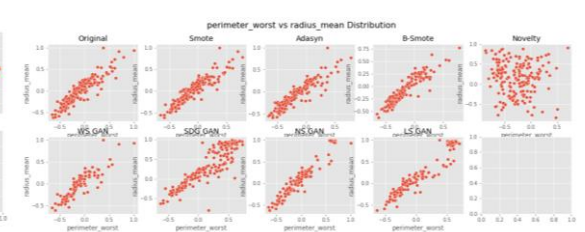
(e)



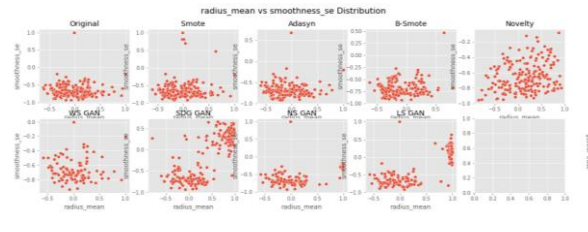
(f)



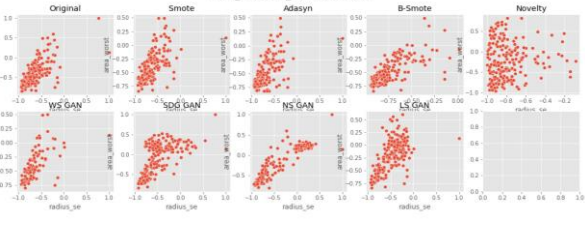
(g)



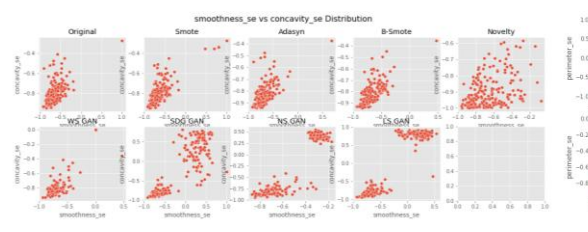
(h)



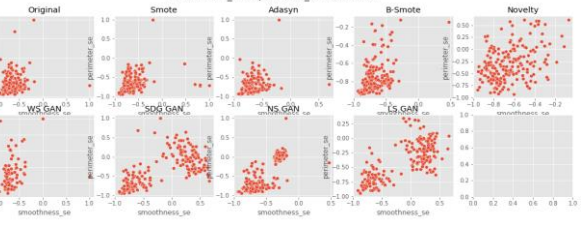
(i)



(j)



(k)



(l)

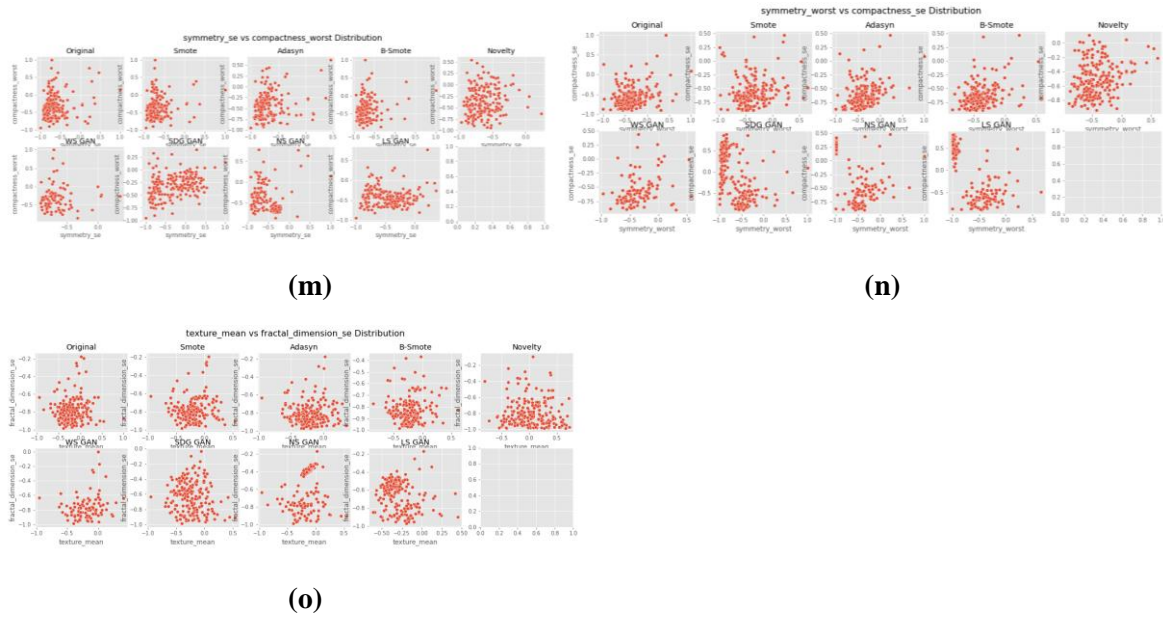


Figure 26: Bi-variate distribution ‘area_mean vs symmetry_worst’ comparison of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla GAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN on breast cancer data (a), ‘compactness_worst vs symmetry_worst’ (b), ‘concavity_se vs symmetry_se’ (c), ‘fractal_dimension_worst vs area_se’ (d), ‘fractal_dimension_worst vs radius_se’ (e), ‘fractal_dimension_worst vs symmetry_worst’ (f), ‘perimeter_se’ vs concavity_mean’ (g), ‘perimeter_worst vs radius_mean’ (h), ‘radius_mean vs smoothness_se’ (i), ‘radius_se vs area_worst’ (j), ‘smoothness_se vs concavity_se’ (k), ‘smoothness_se vs perimeter_se’ (l), ‘symmetry_se vs compactness_worst’ (m), ‘symmetry_worst vs compactness_se’ (n), ‘texture_mean vs fractal_dimension_se’(o)

Figure 26 (a) presents the Bi-variate distribution ‘area_mean vs symmetry_worst’ comparison between the original dataset and SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with breast cancer data. The bivariate visualization charts of data points generated by K-CGAN demonstrate strong agreement with the original dataset, indicating that this algorithm is effective in creating realistic synthetic datasets. This suggests that K-CGAN has great potential for being implemented in future research projects dealing with data augmentation. It is also worth noting that the other algorithms tested performed well, as can be seen from the visualizations. However, K-CGAN stands out among them due to its more stable accuracy in producing samples that closely resemble the original dataset. The effectiveness of the K-CGAN algorithm can be seen from its bivariate visualization charts, which show a high resemblance to the original dataset. Figure 26 (b) compares ‘compactness_worst’ and ‘symmetry_worst’ features and shows that the data points generated by K-CGAN closely match that of the original dataset. This indicates that K-CGAN is effective in generating synthetic datasets, the accuracy of the data points generated by K-CGAN can be attributed to its use of custom hyperparameter architecture and custom loss for generating datasets which accurately reflects the characteristics of the original dataset.

Figure 26 (c) shows the Bi-variate distribution ‘concavity_se vs symmetry_se’ comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with the breast cancer data. The bivariate visualization charts clearly show K-CGAN's capability in generating synthetic datasets that accurately represent the features of the original dataset. The data points generated by the K-CGAN are similar to that of the original dataset, as can be seen from the bivariate visualizations. Figure 26 (d) demonstrates that this approach is effective in generating synthetic ‘fractal_dimension_worst vs area_se’ samples with high similarity to real world data. These findings give credence to the theory that K-CGAN can successfully generate meaningful and useful biases, allowing its

applications to be extended beyond the domain of breast cancer data. The high degree of similarity between these synthetic samples and real world data suggests that K-CGAN can be a powerful tool in augmenting existing datasets for further research.

The bivariate visualizations of the data points generated by K-CGAN illustrate a strong correlation to the original dataset. Figure 26 (e) shows that despite some level of deviation K-CGAN was able to replicate the distribution and variance of the 'fractal_dimension_worst vs radius_se' original data samples, even when faced with imbalanced data. This demonstrates that K-CGAN is an effective algorithm for generating synthetic data. The effectiveness of the K-CGAN algorithm is evident from Figure 26 (f), as it successfully generated data points that are highly comparable to those in the original dataset. This can be seen in particular when comparing the bivariate visualization distribution chart comparison of 'fractal_dimension_worst vs symmetry_worst' data points generated by K-CGAN with those of the original dataset, and other methods. The set of data points generated by K-CGAN show a high degree of similarity to the original dataset, strongly indicating that K-CGAN is an effective algorithm for generating new data. Moreover, due to its highly accurate reproduction of original data samples, K-CGAN could prove invaluable in numerous research and development projects where large amounts of data must be swiftly generated.

K-CGAN is an effective algorithm for generating synthetic data points that closely resemble the real data samples. This is evident from Figure 26 (g) that shows the bi-variate distribution chart comparing 'perimeter_se vs concavity_mean' of original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla CGAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN algorithm offers a reliable way of generating additional samples. In Figure 25 (h), the bivariate visualisation chart comparing 'perimeter_worst vs radius_mean' data points generated by the K-CGAN algorithm showed some deviation. As shown in Figure 25 (i), the bivariate visualization charts generated by the K-CGAN algorithm provide a clear indication of its effectiveness in accurately representing the original dataset. This is a testament to the robustness of the algorithm, showing that it can handle datasets with various complexities without compromising accuracy or precision.

As shown in Figure 26 (j), the bivariate visualization charts of data points generated by K-CGAN have demonstrated agreement with the original dataset for 'radius_se vs area_worst' in presence of some deviations. The visualizations highlight the potential of K-CGAN as a reliable and viable tool for medical data augmentation. The effectiveness of the algorithms was determined by assessing the bi-variate distribution comparison of the original dataset, SMOTE, ADASYN, B-SMOTE, Vanilla GAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN with breast cancer data. As seen from Figure 26 (k), the data points 'smoothness_se vs concavity_se' generated by the K-CGAN prove its quality performance compared to other algorithms.

The effectiveness of the K-CGAN algorithm is evident in the bivariate visualization charts, which demonstrate high agreement with the original dataset. The data points 'smoothness_se vs perimeter_se' generated by K-CGAN show a close resemblance to those present in the original dataset, as can be seen from Figure 26 (l). Furthermore, overall the smoothness and perimeter of the data points generated by K-CGAN were shown to be similar to those from the original dataset, further illustrating its effectiveness in accurately replicating breast cancer data points. Figure 26 (m) shows the bivariate visualization chart comparing 'symmetry_se vs compactness_worst' data points generated by K-CGAN and other methods. The findings suggest that K-CGAN is highly effective in creating synthetic data points which closely resemble the distribution and characteristics of the original dataset. This indicates that K-CGAN is a reliable algorithm for generating new samples which can be used for various applications such as anomaly detection, classification, and regression tasks.

Figure 26 (n) shows that K-CGAN had a high degree of success in preserving the characteristics and properties of the original dataset. These results indicate that K-CGAN has a clear advantage over existing

data augmentation algorithms in terms of accuracy and effectiveness, making it an ideal choice for tackling challenging data augmentation tasks. Figure 26 (o) illustrates visualizations of the bivariate distribution between texture_mean and fractal_dimension_se illustrate a comparison of the original dataset with SMOTE, ADASYN, B-SMOTE, Vanilla GAN, WGAN, SDG GAN, NS GAN, LS GAN and K-CGAN on breast cancer data. It can be seen from the visualizations that overall the data points generated by K-CGAN highly resemble those of the original dataset. This indicates that K-CGAN is an effective algorithm for generating synthetic data with properties similar to those of real data.

4.7.3 Classification performance with original breast cancer data

In the analysis of classification performance using the original imbalanced dataset, a typical 80-20 split was employed for training and testing. 80% of the dataset was used for model training, and the remaining 20% was dedicated to evaluating their performance. This approach ensured a thorough evaluation of the models' ability to generalise and make accurate predictions on unseen data. Table 52 displays the classification performance on the original imbalanced dataset.

Table 52: Classification methods on original imbalanced breast cancer data

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.946429	0.981481	0.963636	0.972028
Random Forest	0.96	0.888889	0.923077	0.944056
Nearest Neighbor	0.977273	0.796296	0.877551	0.916084
MLP	0.962963	0.962963	0.962963	0.972028
Logistic Regression	1	0.962963	0.981132	0.986014

An analysis of the performance indicators of various classifiers on the initial imbalanced breast cancer dataset, as depicted in Table 52, provides a valuable narrative. The performance of the XGBoost classifier demonstrated a satisfactory balance. The model attained a precision value of 0.946429 and a recall value of 0.981481, resulting in an F1 score of 0.963636. The XGBoost model achieved a notable accuracy of 0.972028. Fundamentally, XGBoost exhibited a notable attribute in this context, as it demonstrated a high level of precision and particularly excelled in recall. This suggests that XGBoost was proficient in accurately detecting positive instances within the given dataset. In contrast, the Random Forest algorithm demonstrated a high precision rate of 0.96. Nevertheless, the recall of the model was 0.888889, which was comparatively lower in comparison to XGBoost. As a result, the F1 score of the model reached 0.923077, accompanied with an overall accuracy of 0.944056. This implies that although Random Forest shown competence in accurately predicting positive instances, it may have overlooked certain positive examples, resulting in a slightly lower recall rate. The performance of the Nearest Neighbour classifier was characterised by a precision of 0.977273, indicating a high level of accuracy in correctly identifying positive instances. However, the recall value of 0.796296 was observed to be comparatively lower, suggesting that the classifier had a relatively higher rate of FNs, failing to identify some positive instances. As a result, the F1 score obtained was 0.877551, and the accuracy achieved was 0.916084. The observed disparity between the precision and recall metrics suggests that the Nearest Neighbour algorithm exhibited a high level of precision, but had difficulties in accurately identifying all positive instances. In comparison, MLP exhibited a harmonious equilibrium between precision and recall, with both metrics achieving a value of 0.962963. The

equilibrium state yielded an F1 score of 0.962963 and an accuracy of 0.972028. This finding suggests that MLP successfully maintained high levels of precision and recall, resulting in a consistent performance across several evaluation parameters. Finally, Logistic Regression emerged as an exceptional performer. The model achieved an impressive F1 score of 0.981132, with a precision of 1 and a recall of 0.962963. The classifier had the highest level of accuracy, measuring at 0.986014. This finding suggests that Logistic Regression demonstrated high precision in correctly identifying positive occurrences, while also successfully capturing a large proportion of the positive cases. In summary, the initial dataset pertaining to imbalanced breast cancer exhibited diverse performance patterns among different classifiers. While many algorithms, such as XGBoost and MLP, demonstrated a commendable balance between precision and recall, others, such as Random Forest and Nearest Neighbour, exhibited a modest discrepancy between the two metrics. However, Logistic Regression demonstrated an optimal balance between precision and recall, making it the most effective classifier for this specific dataset.

4.7.4 Classification performance with balanced dataset

In the evaluation of classification performance using the balanced dataset through the application of the K-CGAN model, a similar approach was adopted. An 80-20 split was utilised, allocating 80% of the balanced dataset for model training and reserving the remaining 20% for testing. This strategy ensured that the model's performance was assessed in the context of balanced class distributions, addressing the challenges posed by class imbalance.

Table 53: Classification performance with balanced breast cancer dataset using Novelty K-CGAN oversampling minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.982801	0.991326	0.991326	0.99125
Random Forest	0.977995	0.988875	0.988875	0.98875
Nearest Neighbor	0.982801	0.991326	0.991326	0.99125
MLP	0.985222	0.992556	0.992556	0.99250
Logistic Regression	0.977995	0.988875	0.988875	0.98875

After conducting a comparative analysis of the performance metrics of various classifiers on the balanced breast cancer dataset, following the application of the K-CGAN method to oversample the minority class, the findings presented in Table 53 reveal noteworthy observations and trends when juxtaposed with the performances of the same classifiers on the original imbalanced dataset, as documented in Table 52. The XGBoost model shown a significant improvement in all performance indicators when applied to the balanced dataset. The precision of the system increased dramatically from 0.946429 to 0.982801, while the recall also improved, rising from 0.981481 to an almost flawless 0.991326. The aforementioned enhancement resulted

in a notable increase in the F1 score, rising from 0.963636 to 0.991326. This signifies a more optimal equilibrium between precision and recall. Furthermore, there was a noticeable and statistically significant enhancement in the precision of the XGBoost model, with the accuracy score increasing from 0.972028 to 0.99125. When considering the Random Forest classifier, it becomes apparent that the act of balancing the dataset has also yielded a favourable outcome. The precision of the Random Forest algorithm shown a slight improvement, increasing from 0.96 to 0.977995. Additionally, the recall rate had a significant rise, rising from 0.888889 to 0.988875. The notable increase in recall made a significant contribution to the improvement of the F1 score, which rose from 0.923077 to 0.988875. The model's accuracy exhibited an increase from 0.944056 to 0.98875. Upon examination of the Nearest Neighbour classifier, it is evident that the data exhibits a significant improvement in its performance. The precision increased somewhat from 0.977273 to 0.982801, while the recall demonstrated a significant improvement, rising from 0.796296 to 0.991326. The improvement in recall dramatically reduced the previously reported disparity between precision and recall, leading to a more equitable and elevated F1 score of 0.991326, which represents a substantial increase from 0.877551. The model's accuracy demonstrated significant improvement, rising from 0.916084 to 0.99125. Significant enhancements were observed in both precision and recall metrics for the MLP classifier following the application of dataset balancing techniques. The precision metric exhibited an improvement from 0.962963 to 0.985222, and the recall metric also demonstrated an increase from 0.962963 to 0.992556. The consistent growth observed in both parameters resulted in a higher F1 score of 0.992556, which represents a significant improvement compared to the previous score of 0.962963. In a similar vein, the accuracy of this model was also improved, achieving a value of 0.99250 compared to the previous value of 0.972028. Furthermore, the Logistic Regression model exhibited improved performance when applied to the balanced dataset, despite its already good performance in the imbalanced dataset. The precision shown a marginal decline from an ideal value of 1 to 0.977995, but the recall demonstrated an improvement from 0.962963 to 0.988875. Consequently, the F1 score exhibited an increase from 0.981132 to 0.988875, suggesting a heightened level of equilibrium in performance. The model's accuracy demonstrated a slight yet favourable increase, progressing from 0.986014 to 0.98875. In conclusion, it is evident that the utilisation of the K-CGAN approach resulted in the generation of a balanced breast cancer dataset, which subsequently contributed to enhanced and consistent performances across all classifiers. This highlights the significance of mitigating class disparities in datasets, particularly as exemplified by instances such as the Nearest Neighbour classifier, which exhibited a notable improvement in the balance between precision and recall when applied to a dataset that was appropriately balanced.

Furthermore, the ROC curves exhibited enhanced performance, with the AUC values consistently increasing after the oversampling technique was applied. It is worth noting that the impact of oversampling on the performance of different classifiers varied, yet all models achieved remarkable results. Particularly, XGBoost, Nearest Neighbor, and MLP demonstrated exceptional precision, recall, F1 score, and accuracy. Additionally, Random Forest and Logistic Regression also exhibited robust performance across these evaluation metrics.

The utilisation of oversampling, specifically with the novelty K-CGAN, facilitated the creation of more diverse synthetic samples, which better represented the underlying distribution of the minority class. This, in turn, aided in reducing overfitting and improving the generalisation capability of the models to unseen data. Moreover, the analysis of the generator and discriminator losses revealed the effectiveness of the novelty K-CGAN in converging to a stable solution. As the training progressed, the generator gradually learned to generate realistic synthetic samples, while the discriminator became increasingly proficient at distinguishing between real and fake data. Overall, the combination of oversampling with the novelty K-

CGAN technique proved to be successful in addressing the class imbalance challenge, enhancing the performance of the classifiers, and improving the overall robustness and generalisation ability of the models.

4.7.5 Classification performance with balanced dataset multiple models comparison

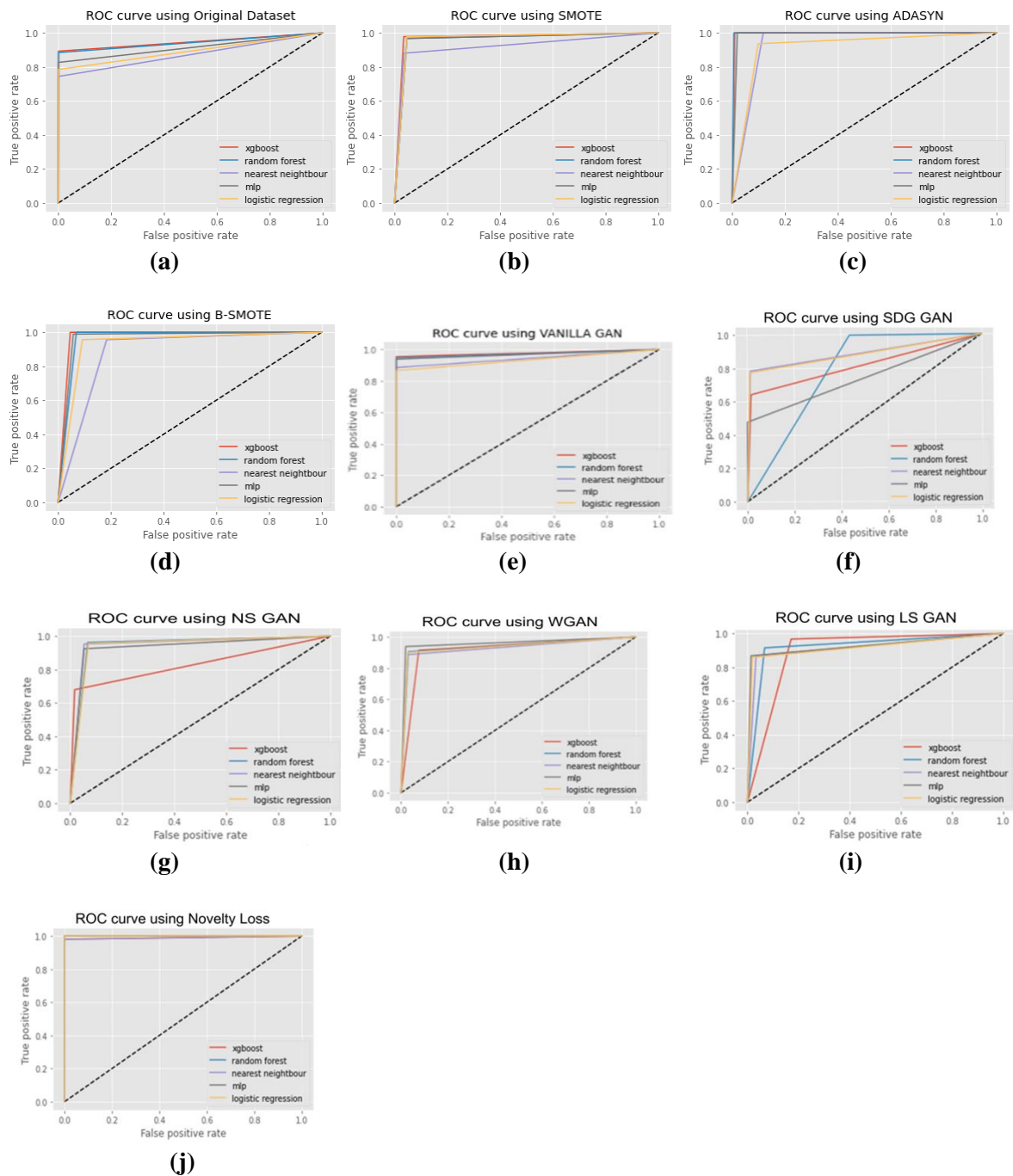


Figure 27: ROC curves (a) original imbalanced dataset, (b) balanced dataset with SMOTE, (c) balanced dataset ADASYN, (d) balanced dataset with B-SMOTE, (e) balanced dataset with Vanilla CGAN, (f) balanced dataset with SDG GAN, (g) balanced dataset with NS GAN, (h) balanced dataset with WGAN, (i) balanced dataset with LS GAN, (j) balanced dataset with K-CGAN

The ROC curves demonstrated a significant improvement in performance when comparing imbalanced and balanced datasets are shown in Figure 27. After applying various oversampling techniques, the AUC values consistently increased, indicating the effectiveness of these methods. However, it is important to highlight that the impact of oversampling on the performance of different classifiers varied, showcasing the nuanced nature of the results. Despite these variations, all models achieved remarkable outcomes, further validating the benefits of employing oversampling in addressing class imbalance.

Comparison of all classification methods and models

Table 54: F1 Score values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.857741	0.896074	0.79558	0.768362	0.963636	0.991326	0.972973	0.956522	0.978723	0.971751
Random Forest	0.902326	0.925301	0.927273	0.728843	0.923077	0.988875	0.962162	0.956989	0.968421	0.965909
Nearest Neighbour	0.901961	0.917073	0.933025	0.868841	0.877551	0.991326	0.920455	0.91579	0.897959	0.934132
MLP	0.917706	0.952153	0.918033	0.645367	0.962963	0.992556	0.962162	0.951351	0.968085	0.965909
Logistic Regression	0.907731	0.924939	0.922727	0.863158	0.981132	0.988875	0.967742	0.951872	0.93617	0.99422

Upon examination of the F1 scores obtained from the application of different oversampling methods to the breast cancer dataset, the Table 54 summarizes the findings. The F1 score is a mathematical measure that combines precision and recall, providing a comprehensive evaluation statistic. A higher score approaching 1 indicates superior performance in terms of combined precision and recall. In the context of the original imbalanced breast cancer dataset, the Logistic Regression model demonstrated the best F1 score, roughly 0.981, with XGBoost and MLP closely trailing behind. It is noteworthy that the use of K-CGAN oversampling resulted in a significant increase in F1 scores for all classifiers. The MLP classifier had the highest F1 score, approximately 0.993, while XGBoost and Nearest Neighbour closely followed with a score of 0.991326. Upon examining the GAN-based methodologies, it is evident that the performance outcomes are varied. The utilisation of LS GAN and WGAN techniques typically resulted in classifiers attaining F1 scores ranging from 0.85 to 0.95. However, the implementation of SDG GAN unexpectedly lowered the efficacy of the classifiers, as evidenced by the significant decline of MLP's F1 score to 0.645. The outcomes of NS GAN, specifically in relation to Nearest Neighbour (about 0.933) and Random Forest (around 0.927), demonstrate noteworthy performance. These findings indicate that the oversampling technique employed by NS GAN could potentially enhance the effectiveness of these classifiers. The performance of the VANILLA CGAN was marginally inferior to that of K-CGAN, however it yielded comparable F1 scores. Notably, in the context of Logistic Regression, the VANILLA CGAN achieved an excellent score of around 0.994. Based on the conducted investigation, it is apparent that the utilisation of the K-CGAN oversampling technique yields a substantial and favourable influence on the performance of classifiers, particularly in the case of MLP, XGBoost and Nearest Neighbour. When selecting an oversampling strategy for the breast cancer data, it is reasonable to use K-CGAN due to its constant improvement across many evaluation metrics. However, it is important to consider the selection of a classifier while making this decision, as the effectiveness of the

oversampling approach can vary depending on the compatibility between the classifier and the chosen method.

Table 55: Accuracy Score values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.8804 91	0.92091 4	0.869947	0.855888	0.972028	0.99125	0.972067	0.955556	0.977654	0.972067
Random Forest	0.9261 86	0.94551 8	0.943761	0.724077	0.944056	0.98875	0.960894	0.955556	0.968421	0.96648
Nearest Neighbour	0.9297 01	0.94024 8	0.949033	0.910369	0.916084	0.99125	0.921788	0.911111	0.897959	0.938547
MLP	0.9420 03	0.96485 1	0.988489	0.804921	0.972028	0.9925	0.960894	0.95	0.968085	0.96648
Logistic Regression	0.9349 74	0.94551 8	0.940246	0.908612	0.986014	0.98875	0.96648	0.956989	0.93617	0.994413

By examining the accuracy ratings of different classifiers over a range of oversampling techniques, insights into the efficacy of each combination of methods in accurately classifying the breast cancer dataset is presented in Table 55. After doing an evaluation of the original imbalanced breast cancer dataset, it was evident that Logistic Regression has notable performance with an accuracy of roughly 0.986. This is a comparatively high score to the performance of XGBoost, which achieved an accuracy of around 0.972. The incorporation of K-CGAN oversampling into the data results in a notable increase in accuracy for all classifiers. Among them, the MLP classifier demonstrates the highest accuracy at approximately 0.9925, while XGBoost and Nearest Neighbour classifiers closely trail behind, achieving approximately 0.99125 accuracy. Upon examining the GAN-based methodologies, it becomes evident that WGAN demonstrates the highest level of consistent improvement across classifiers. Specifically, the MLP classifier achieves an accuracy of around 0.96485, while the Random Forest classifier achieves an accuracy of approximately 0.94551. In sharp contrast, the utilisation of SDG GAN leads to a notable loss in performance, particularly seen in the case of Random Forest, where the accuracy drops significantly to 0.724077. The performance of the NS GAN model exhibits remarkable potential, particularly when combined with MLP architecture, yielding an accuracy level of roughly 0.98849. The accuracy values obtained with VANILLA CGAN are comparable to those of K-CGAN, particularly when considering Logistic Regression. Notably, Logistic Regression achieves an impressive accuracy rate of around 0.99441. This comprehensive analysis emphasises the effectiveness of the K-CGAN oversampling technique, which regularly enhances classifier performance by significantly improving accuracy scores. However, it is important to examine the alignment between the classifier and the oversampling technique when selecting a strategy for the breast cancer dataset, as the benefits can differ.

Table 56: Recall Score values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.966981	0.915094	0.679245	0.641509	0.981481	0.991326	0.97826 1	0.946237	1	0.988506
Random Forest	0.915094	0.90566	0.962264	0.995283	0.888889	0.988875	0.96739 1	0.956989	1	0.977012
Nearest Neighbour	0.867925	0.886792	0.95283	0.783019	0.796296	0.991326	0.88043 5	0.935484	0.956522	0.896552
MLP	0.867925	0.938679	0.924528	0.476415	0.962963	0.992556	0.96739 1	0.946237	0.98913	0.977012
Logistic Regression	0.858491	0.900943	0.957547	0.773585	0.962963	0.988875	0.97826 1	0.956989	0.956522	0.988506

Table 56 shows that in the original imbalanced breast cancer dataset, XGBoost demonstrated a notable recall value of 0.981481. This performance was marginally surpassed when the K-CGAN oversampling technique was employed in conjunction with XGBoost, resulting in a recall score of 0.991326. This suggests that the model demonstrated a high level of accuracy in accurately recognising the positive cases within the dataset. The WGAN demonstrated a varied tendency to enhance the recall performance of some classifiers, while at the same time lowered for some classifiers such as for XGBoost, MLP and Logistic Regression. In the case of XGBoost, the recall value was reduced to be 0.915094 from 0.981481, however with the Nearest Neighbour, the recall value increased to 0.886792 from 0.796296. In contrast, the utilisation of the SDG GAN approach appeared to have a negative impact on the recall scores, particularly in the case of XGBoost and MLP, where the respective values were seen to be 0.641509 and 0.476415. The performance of NS GAN was varied, with notable improvements in recall observed when combined with Random Forest, achieving a score of 0.962264. However, when applied with XGBoost, it resulted in a decrease in recall to 0.679245. B-SMOTE, one of the conventional oversampling techniques, consistently improved the recall values. It achieved a flawless score of 1 when applied to both XGBoost and Random Forest algorithms. ADASYN, B-SMOTE and SMOTE shown an increase in recall scores for most classifiers. Specifically, SMOTE showed a strong synergy with Random Forest, Nearest Neighbour, MLP and Logistic Regression resulting in recall scores of 0.967391, 0.880435, 0.967391, and 0.978261 respectively. While B-SMOTE achieving 1 for XGBoost and Random Forest, and 0.98913 with MLP. The effectiveness of VANILLA CGAN, similar to K-CGAN, was demonstrated across many classifiers, notably enhancing the recall scores for XGBoost and Logistic Regression. In both cases, the recall scores reached a value of 0.988506. In conclusion, the K-CGAN oversampling technique has been identified as a highly effective approach for enhancing recall performance in classifiers. However, it is important to carefully analyse the compatibility between the classifier and oversampling method while choosing a strategy for the breast cancer dataset, as the effectiveness displayed exhibited subtle differences.

Table 57: Precision Score values for classification methods multiple methods comparison

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.770677	0.877828	0.96	0.957747	0.946429	0.982801	0.966981	0.946237	0.978723	0.955556
Random Forest	0.889908	0.945813	0.894737	0.574932	0.96	0.977995	0.915094	0.956989	0.968421	0.955056
Nearest Neighbour	0.938776	0.949495	0.914027	0.97078	0.977273	0.982801	0.867925	0.935484	0.897959	0.975
MLP	0.973545	0.966019	0.911628	1	0.962963	0.985222	0.867925	0.946237	0.98913	0.955056
Logistic Regression	0.962963	0.950219	0.890351	0.97619	1	0.977995	0.858491	0.956989	0.956522	1

Through an investigation into the performance of oversampling strategies, it was shown that the K-CGAN method had a notable efficacy in significantly improving precision across all domains as per Table 57. The MLP model achieved the highest precision value of 0.985222, while the XGBoost model closely followed with a precision value of 0.982801. It is worth mentioning that the Nearest Neighbour classifier demonstrated a commendable precision score of 0.982801 when used in conjunction with K-CGAN. The integration of classifiers with WGAN resulted in a significant enhancement in precision across the majority of models. As an example, the precision of XGBoost experienced a substantial decline, reaching a value of 0.877828 from 0.946429, whereas Random Forest exhibited a notable decline, achieving a precision of 0.945813. The approach of SDG GAN demonstrated varied results. Although it led to a precise measurement of 1 for MLP, it significantly reduced the precision for Random Forest, yielding a score of 0.574932. Various traditional oversampling approaches, such as SMOTE, ADASYN, and B-SMOTE, have demonstrated diverse impacts on precision. The B-SMOTE technique exhibited strong compatibility with the XGBoost algorithm, resulting in a notable increase in precision to 0.978723. K-CGAN resulted in the second highest MLP 0.98913 rising from original 0.962963. In a similar vein, it was shown that ADASYN had a favourable compatibility with both Random Forest and MLP, resulting in precision scores surpassing 0.946. The logistic regression original value of 1 was reduced by all models except Vanilla CGAN. The Vanilla CGAN approach consistently enabled the achievement of high precision levels across several classifiers. The Logistic Regression model achieved a precision of 1, indicating a high level of accuracy. On the other hand, the Nearest Neighbour model achieved a score of 0.975, suggesting a strong performance. In conclusion, the utilisation of the K-CGAN oversampling method consistently enhanced precision across various classifiers. However, the findings also underscored the importance of the interplay between individual classifiers and oversampling strategies. The outcomes of each pairing exhibited distinct nuances, underscoring the importance of meticulous deliberation while striving to optimise precision for the breast cancer dataset.

Chapter 5

Discussion

This chapter further discusses the key experimental results. The results of the research will serve as a valuable insight for future explorations into the capabilities of GANs. The experimental results of applying the K-CGAN method to credit card fraud data have proven that it is capable of generating high quality datasets with stable training and improved classification performance. The optimized hyperparameter architecture was able to deliver an effective credit card fraud detection method using data augmentation. Our experiments also demonstrated the introduction of a custom loss function that incorporates KL divergence loss to ensure both distributions are close to each other and custom optimised hyperparameter architecture. This approach was able to successfully improve the quality of the synthetic data generated by K-CGAN and achieve better performance compared to other GAN-based architectures. Additionally, this model was further tested and trained on the breast cancer dataset, with optimised hyperparameters achieving superior performance. The results demonstrate that K-CGAN can generalise to other datasets and be used effectively for data augmentation. This method has shown promising results in its ability to improve classification performance while maintaining a stable training process and sample distribution. As future work, further research could be conducted into the application of K-CGAN to other domains and datasets. Additionally, exploring further optimization techniques could be conducted in order to improve the performance of this model. Through these experiments, a better understanding can be gained into when and how such methods should best be applied for various applications.

Discussion on Novelty Loss GAN Experiments

The experimental results with KL loss indicate that the addition of KL loss greatly improved the cosine similarity values across all variables in comparison to without KL loss. This improvement demonstrates the enhanced quality of the synthetic data generated by the K-CGAN model. The absence of KL loss could explain the lower cosine similarity value observed. The inclusion of KL loss serves as a regularizer, ensuring that the generator produces samples that align with the distribution of the training data, thereby preventing overfitting. Removing this term may have allowed the generator to overfit to the training data, potentially compromising the quality of the generated samples. Mode collapse, where the generator produces a limited number of samples that fail to accurately represent the underlying data distribution, can occur when the generator lacks the encouragement to provide sample diversity, as offered by the KL loss term. Insufficient variation in the training data can also hinder the generator's ability to learn to produce samples that align with the data distribution. The removal of the KL loss term may exacerbate this issue, leading to a decline in cosine similarity. Moreover, the removal of the KL loss component may have affected the optimal custom hyperparameters for the generator, further impacting the quality of the generated samples. Additionally, the

absence of the KL loss term may increase the stochasticity of the training process, resulting in lower sample quality, as GAN training relies on a certain level of unpredictability.

The study used the SMOTE method to oversample the minority class (fraudulent transactions) to balance the dataset, we oversampled the minority class instances of fraudulent transactions to ensure sufficient representation for both classes in the dataset used for GAN training. This step was crucial because models trained on imbalanced datasets often exhibit poor generalisation performance when faced with unseen data. During the training phase, we utilised the entire balanced dataset to ensure comprehensive coverage of the data nuances. No additional preprocessing techniques were necessary as the dataset had already undergone meticulous cleaning and labelling. However, once the GAN is trained, SMOTE is no longer required for generating synthetic samples. Furthermore in this experiment we have utilized our custom K-CGAN method. The cosine similarity analysis of experiments confirmed that the cosine similarity scores between the real and fake datasets for all features have considerably enhanced after presenting SMOTE oversampling method along with K-CGAN before training.

Experiments with batch normalisation

In this set of experiments we evaluated the impact of batch normalisation on GANs training. The use of batch normalisation appears to be an effective way to promote stability and generate more varied samples. Additionally, the range in discriminator loss demonstrates the model's ability to accurately identify original samples from generated ones. The generator loss was extremely unstable at initial phase ranging from 0.6 to 0.8. On the other hand, discriminator loss ranged between 0.4 and 1.0. However, the training procedure was stabilised and the model was kept out of local minima by using batch normalisation. The introduction of batch normalisation has made generator and discriminator losses more stable and predictable. The variety in the discriminator loss indicates the ability in differentiating between actual and synthetic data, while the greater initial generator loss shows that the model accurately generates more varied samples right away.

The performance of a GAN-based novelty detection system can be examined by looking at the loss values of both the generator and discriminator networks. Generally, we would expect to see an increase in the discriminator loss over time, as it is becoming better at distinguishing real from synthetic data. On the other hand, the generator's loss should decrease since it is getting better at producing realistic samples. The variety in the discriminator loss shows that the model is successfully differentiating between actual and synthetic data, while the greater initial generator loss may signal that the model generates more varied samples right away.

It is also important to note that training a GAN-based method can be computationally expensive, due to its two neural network architecture. Therefore, careful tuning of the model parameters and architecture is essential in order to get the best performance out of the system. Additionally, since GANs are generative models, they can be used to generate synthetic data that can be used for further training or testing of other machine learning systems. This makes GAN-based data augmentation a powerful tool in a variety of applications.

Discussion on GAN Hyperparameter tuning with credit card fraud and breast cancer data

This section offers a detailed discussion of GAN hyperparameter tuning with credit card fraud data. A total of 1152 experiments using the grid search method (Bergstra and Bengio, 2012). The experiments intend to identify the optimal values for hyperparameters in order to generate realistic credit card fraud data. In order to achieve this, several variables must be taken into account, including the number of hidden layers and weights initialization technique used in the neural network (Rumelhart et al., 1986; Glorot and Bengio, 2010). Additionally, two more hyperparameters are essential for training accuracy: the learning rate and

dropout rate. In order to assess how different combinations of hyperparameter values affect the model, the experiment must be conducted several times with various configurations. This process can be laborious and time-consuming; however, it is a necessary step in order to obtain an accurate model with a good predictive power. Different techniques such as grid search, random search, or evolutionary search can be used to identify the optimal combination of hyperparameters. Once the best values for each hyperparameter have been determined, a final evaluation of the model must be conducted in order to determine its accuracy and reliability. It is imperative to mention that the potential combinations of hyperparameters can be overwhelming, and the optimal values for each will vary depending on the desired output. To determine which hyperparameter settings are most effective for a particular task, accurate experimentation is necessary. This may include testing multiple configurations including variations in batch size, learning rate, activation functions, layers of both discriminator and generator networks as well as the various optimization techniques such as Adam, RMSprop and Adagrad.

Our experiments concluded the selection of hyperparameters significantly affects the performance of a deep learning model. As such, it is important to understand which combinations are most effective in order to maximise training and prediction accuracy. In this section, we discussed the top five combinations of hyperparameters based on their loss values, as well as notable trends that can be observed. The best-performing hyperparameter combinations frequently use LeakyReLU as their activation functions due to its ability to avoid saturation and allow for a more consistent learning. Furthermore, the larger 32 batch size appears to perform well in all combinations, presumably as a result of improved training efficiency and stability. Additionally, the best-performing combinations all utilise a different optimizer, with Adam appearing in three of the top five, followed by Adagrad and RMSprop.

The optimised K-CGAN method for credit card fraud data, Generator neural network is composed of ReLU activation and trained using a combination of modified binary cross entropy and KL divergence. The Generator network has three layers (2 hidden layers, 1 output layer) with 64, 32, 29 neurons. The network incorporates a dropout rate of 0.1 and employs Adam optimizer with a learning rate of 0.0001. Additionally, it utilizes a 100-dimensional noise vector and applies glorot_uniform kernel initialization with L2 regularization. With a total of 36,837 parameters, the generator network is capable of generating high-quality samples. On the other hand, the K-CGAN Discriminator neural network utilizes Leaky ReLU activation and binary cross entropy loss for binary classification. The Discriminator network has three layers (2 hidden, 1 output layer) with 20, 15 and 1 neurons respectively. Similar to the generator network, Discriminator also incorporates a dropout rate of 0.1 and uses Adam optimizer with a learning rate of 0.0001. L2 regularization is applied to prevent overfitting. With a total of 1,519 parameters, the Discriminator network effectively distinguishes between real and generated data, thereby contributing to the K-CGAN model's ability to generate realistic samples.

The K-CGAN model is specifically designed for the breast cancer dataset, aiming to generate synthetic data that closely resembles real breast cancer data. The generator neural network, a crucial component of the K-CGAN model, consists of three layers (2 hidden layers, 1 output layer) with 64, 32, 29 neurons. The activation function used in the generator neural network is ReLU, known for its ability to handle non-linearities effectively. To optimize the generator's performance, the loss function combines the modified binary cross entropy with KL divergence. The output optimizer employed is Adam, a popular algorithm that adapts learning rates for each parameter. In this case, the learning rate is set to 0.0001, ensuring smooth convergence. To introduce some regularization, a dropout layer is included in the generator neural network with a dropout rate of 0.2. Total of 10,226 learning parameters. On the other hand, the discriminator neural network plays a crucial role in distinguishing between real and synthetic breast cancer data generated by the generator. Similar to the generator, the discriminator consists of three layers (2 hidden, 1 output layer) with 20, 15 and 1 neurons respectively. The activation function used in the discriminator neural network is LeakyReLU, which helps to alleviate the vanishing gradient problem by allowing small negative values. The

loss function employed in the discriminator neural network is binary cross-entropy, which measures the dissimilarity between predicted and target labels. Similar to the generator, the discriminator utilizes Adam optimizer with a learning rate of 0.0001. Additionally, a dropout layer is added to the discriminator neural network with a dropout rate of 0.2, enhancing its generalization capability. With a total of 1,386 learning parameters, by carefully tuning these hyperparameters and leveraging the power of the K-CGAN model, we can effectively generate synthetic breast cancer data that holds great potential for various research and analysis applications.

Discussion on Classification Performance: Imbalanced Original Datasets vs. Balanced Minority Class Oversampled datasets in credit card fraud and breast cancer data

Table 58: Summary of Classifiers performance on original imbalanced credit card fraud data

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.981818	0.827068	0.873016	0.999551
Random Forest	0.981651	0.812030	0.867470	0.999537
Nearest Neighbor	1.000000	0.721804	0.786885	0.999270
MLP	0.990099	0.842105	0.861538	0.999494
Logistic Regression	0.989583	0.609023	0.723214	0.999129

Table 59: Classification performance with balanced credit card fraud data by leveraging K-CGAN for oversampling the minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.997998	0.999706	0.997599	0.9976
Random Forest	0.999598	0.999706	0.997394	0.9974
Nearest Neighbor	0.990056	0.999706	0.992820	0.9928
MLP	0.998400	0.999594	0.998400	0.9984
Logistic Regression	0.991221	0.999608	0.992409	0.9924

Table 60: F1 Score values for classification methods multiple methods comparison credit card fraud data

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.961263	0.960000	0.971519	0.950336	0.873016	0.997599	0.999733	0.999584	0.999760	0.975749
Random Forest	0.956772	0.953168	0.966720	0.951743	0.867470	0.997394	0.999881	0.999880	0.999809	0.969957
Nearest Neighbour	0.931983	0.925414	0.937799	0.925776	0.786885	0.992820	0.991008	0.986707	0.998673	0.921512
MLP	0.950355	0.947658	0.959119	0.942105	0.861538	0.998400	0.998844	0.998949	0.998913	0.960000
Logistic Regression	0.905830	0.909859	0.915584	0.901526	0.723214	0.992409	0.946270	0.884358	0.995553	0.913108

The study involved an analysis of the performance of classifiers on a dataset for credit card fraud detection. The findings of this analysis are subsequently provided in Tables 58, 59, and 60. The efficacy of employing the K-CGAN technique was effectively proved in improving the performance of classifiers for this particular challenge. The K-CGAN successfully achieved dataset balancing by producing synthetic fraud transactions that closely matched the genuine ones. The aforementioned findings were apparent in the outcomes displayed in Tables 59 and 60, whereby a notable enhancement was noticed in the metrics of precision, recall, and F1 scores for all the models. The achievement of remarkable results by the MLP classifier is worth noting. The results indicated an F1 score, precision, and recall of 0.9984, suggesting a strong capacity to detect fraudulent transactions. In a similar vein, the logistic regression classifier, which previously had comparatively lower scores, saw significant enhancement with F1, accuracy, and recall scores all converging around the 0.99 threshold. The significance and efficacy of the K-CGAN-based oversampling technique are highlighted by the observed improvements in performance measures. This is particularly relevant when addressing the imbalances that are frequently seen in fraud detection datasets. The metrics of high precision and recall play a crucial role in the field of fraud detection, as they contribute to the reduction of FPs and the precise identification of legitimate instances of fraud. The impressive performance exhibited by these classifiers, subsequent to the implementation of the K-CGAN method, garners considerable interest in the potential of this methodology. The scope of this issue extends beyond the detection of credit card fraud. Considering the proven effectiveness of K-CGANs, it is plausible to imagine the potential application of this technique in other fields that face challenges related to class imbalances, such as insurance fraud and tax fraud. This work not only confirms the effectiveness of K-CGAN as a method for augmenting data, but also suggests its potential as a solution for improving the performance of models in various fraud detection scenarios. In a comprehensive analysis of Table 60, which presents the F1 score values for various approaches across diverse classifiers, several distinct trends and significant implications can be observed.

The K-CGAN method demonstrated superior performance compared to numerous other strategies over a wide range of classifiers. In the case of the XGBoost, the F1 score achieved with K-CGAN was 0.997599, demonstrating a significant improvement compared to the F1 score of 0.873016 obtained from the original dataset. Similarly, the Random Forest classifier presents a compelling case for K-CGAN. The F1 score, when paired with K-CGAN, registers at 0.997394, a figure substantially higher than the 0.867470 obtained from the original dataset. The difference approximates to a remarkable increase of nearly 12.99%. The MLP classifier also echoes this trend of enhancement with K-CGAN. The model, when trained with K-CGAN augmented data, secures an F1 score of 0.998400, a considerable rise from the 0.861538 score linked with the original dataset. This translates to a significant boost of about 13.69% in performance. Moreover, the F1 score achieved by K-CGAN outperforms other generative adversarial networks such as LS GAN, WGAN,

and SDG GAN. This pattern remains constant across different classifiers as well. In the case of Random Forest, the K-CGAN model demonstrated a significantly higher F1 score of 0.997394 compared to the original dataset's score of 0.867470. It is important to mention that the performance of K-CGAN was comparable to other oversampling approaches such as SMOTE, ADASYN, and B-SMOTE, which have typically been used to address class imbalances. The initial F1 score of the Nearest Neighbour classifier on the original dataset was quite low, measuring 0.786885. However, after applying K-CGAN, there was a substantial improvement in performance, resulting in an F1 score of 0.992820. In a similar vein, the F1 score of the MLP classifier experienced a substantial increase from 0.861538 on the initial dataset to 0.998400 when employing the K-CGAN technique. This outcome further reinforces the significant and profound influence of the K-CGAN strategy.

Moreover, when comparing K-CGAN with other generative models like Vanilla CGAN, it becomes evident that although Vanilla CGAN obtained commendable outcomes, it was generally surpassed by K-CGAN in the majority of classifier comparisons. For example, when employing the XGBoost model, the Vanilla CGAN algorithm attained an F1 score of 0.975749. Although this achievement is noteworthy, it is still inferior to the F1 score obtained by the K-CGAN algorithm. Based on the findings, it can be concluded that K-CGAN successfully tackles the inherent class imbalance in the credit card fraud dataset. Additionally, it enhances the performance of classifiers by providing them with synthetically created data that is both diverse and of high quality. The efficacy and versatility of this approach, in comparison to established methodologies including both conventional and GAN-based techniques, demonstrates its resilience and flexibility. Moreover, the consistent enhancements observed in various classifier types, ranging from ensemble models like XGBoost to neural networks such as MLP, demonstrate the generalizability of K-CGAN and emphasise its potential as a benchmark method for augmenting data in credit card fraud detection and other scenarios involving imbalanced datasets.

Table 61: Summary of Classifiers performance on original imbalanced breast cancer data

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.946429	0.981481	0.963636	0.972028
Random Forest	0.96	0.888889	0.923077	0.944056
Nearest Neighbor	0.977273	0.796296	0.877551	0.916084
MLP	0.962963	0.962963	0.962963	0.972028
Logistic Regression	1	0.962963	0.981132	0.986014

Table 62: Classification performance with balanced breast cancer data using K-CGAN to oversample minority class

Model	Precision	Recall	F1 Score	Accuracy
XGBoost	0.982801	0.991326	0.991326	0.99125
Random Forest	0.977995	0.988875	0.988875	0.98875
Nearest Neighbor	0.982801	0.991326	0.991326	0.99125
MLP	0.985222	0.992556	0.992556	0.99250
Logistic Regression	0.977995	0.988875	0.988875	0.98875

Table 63: F1 Score values for classification methods multiple methods comparison breast cancer data

Model	LS GAN	WGAN	NS GAN	SDG GAN	Original dataset	K-CGAN	SMOTE	ADASYN	B-SMOTE	VANILLA GAN
XGBoost	0.857741	0.896074	0.79558	0.768362	0.963636	0.991326	0.972973	0.956522	0.978723	0.971751
Random Forest	0.902326	0.925301	0.927273	0.728843	0.923077	0.988875	0.962162	0.956989	0.968421	0.965909
Nearest Neighbour	0.901961	0.917073	0.933025	0.868841	0.877551	0.991326	0.920455	0.91579	0.897959	0.934132
MLP	0.917706	0.952153	0.918033	0.645367	0.962963	0.992556	0.962162	0.951351	0.968085	0.965909
Logistic Regression	0.907731	0.924939	0.922727	0.863158	0.981132	0.988875	0.967742	0.951872	0.93617	0.99422

The performance of all classifiers has been enhanced by oversampling the minority class using the novelty K-CGAN as depicted in Tables 62 and 63. When trained on the oversampled dataset, all classifiers have demonstrated significant improvement in F1 score, precision, recall, and accuracy compared to the original dataset. The use of the novelty K-CGAN to generate synthetic data has effectively balanced the class distribution and led to a substantial improvement in classifier performance. The experimental results of applying the K-CGAN method to breast cancer data have shown its capability to generate high-quality data with stable training and improved classification performance. By optimising hyperparameter settings, we have established an effective breast cancer detection method using data augmentation.

While examining Table 61, which presents the outcomes of the initial imbalanced breast cancer dataset, it becomes apparent that the classifiers had a relatively satisfactory performance despite the presence of imbalances. The XGBoost algorithm exhibited strong performance, achieving an F1 score of 0.963636. Logistic Regression demonstrated remarkable performance with a precision of 1 and an F1 score of 0.981132, underscoring its efficacy in accurately detecting malignant tumours, even in the presence of

imbalanced data. Nevertheless, after applying the novelty K-CGAN approach to balance the dataset by oversampling the minority class, as demonstrated in Table 62, a significant improvement in classifier performance was observed. Significantly, precision, recall, and F1 scores shown a notable increase to values exceeding 0.98 for models such as XGBoost, MLP, and Nearest Neighbour. The XGBoost algorithm, for instance, achieved a robust F1 score of 0.991326 improving from 0.963636. Notably, the Logistic Regression stood out with a flawless precision of 1, culminating in an F1 score of 0.981132, a figure that accentuates its prowess in accurately discerning malignant tumors even when faced with imbalanced data. However, the narrative shifts intriguingly upon the application of the K-CGAN approach, a method aimed at counterbalancing the dataset by oversampling the minority class. The Random Forest algorithm, for example, which previously showcased an F1 score of 0.923077 on the original data, exhibited a significant leap to 0.988875 when trained on the K-CGAN enhanced dataset. Similarly, the Nearest Neighbour classifier, which initially returned an F1 score of 0.877551, surged impressively to reach 0.991326 under the influence of K-CGAN. MLP was also improved. Starting with an already impressive F1 score of 0.962963 on the original dataset, the introduction of K-CGAN data took its performance up a notch to a striking F1 score of 0.992556. This demonstrates the capability of K-CGAN to not only achieve dataset balance, but also enhance it in a way that enables classifiers to effectively distinguish between benign and malignant tumours by capturing subtle distinctions.

The exceptional performance, particularly following the use of K-CGAN oversampling, is further emphasised when considering the data presented in Table 63. The study presents the F1 score values for different approaches applied to various classifiers. The K-CGAN approach demonstrated superior performance in terms of F1 scores compared to both the original dataset and alternative approaches such as LS GAN, WGAN, and VANILLA CGAN, over a wide range of classifiers. For example, when utilising the XGBoost model, the F1 score obtained using K-CGAN was 0.991326. This result demonstrates a significant enhancement compared to the original dataset score of 0.963636, as well as surpassing the performance of alternative techniques such as SMOTE and ADASYN. Moreover, the Nearest Neighbour classifier, which achieved an F1 score of 0.877551 on the initial imbalanced dataset, exhibited a substantial improvement to 0.991326 when utilising K-CGAN. In a similar vein, the F1 score of the MLP model exhibited a notable increase from 0.962963 on the original dataset to an impressive 0.992556 when utilising the K-CGAN technique. The efficacy of the K-CGAN strategy is demonstrated by the persistent improvements observed in the classifiers. These improvements are evident not just when compared to the original imbalanced dataset but also in comparison to alternative oversampling methods. The results demonstrate that K-CGAN not only augments the dataset size, but also guarantees that these instances accurately represent the underlying distribution and intricacies, hence enabling classifiers to effectively differentiate malignant tumours with improved precision. The K-CGAN approach was found to be crucial in enhancing classifier performance for breast cancer data, similar to its effectiveness in analysing credit card fraud data. The significant impact of K-CGAN underscores its potential as an essential tool for data scientists dealing with imbalanced datasets in the field of medical diagnostics. In this domain, precision and recall are not only measures, but rather influential elements that can greatly affect patient outcomes.

Chapter 6

Conclusion and Future Work

This chapter summarises the contributions and the achievements, and further recommends future research directions.

Adding the KL divergence to the loss function of K-CGAN and its custom architecture including additional hyperparameters Kernel Initializer ‘gloriot_uniform’ method (Glorot, 2010) for the Generator and Kernel Regularizer ‘L2 method’ (Bishop, 2006) for both Generator and Discriminator has been found to enhance the quality and diversity of the generated samples, provide better command over the data distribution, and make GAN training more stable and dependable. Our experiments showed that this approach could enhance the quality of the synthetic data generated by K-CGAN and outperform other GAN-based architectures. The K-CGAN method was trained on credit card fraud data and the results showed that it can generate high quality datasets with stable training and improved classification performance. By optimising the hyperparameter settings, an effective credit card fraud detection method was developed using data augmentation, resolving data imbalance issues and improving classification accuracy.

Moreover, we trained and tested K-CGAN on the breast cancer dataset and achieved superior performance by optimising its hyperparameters to suit that type of data. The findings show that K-CGAN has the ability to operate with other datasets and can be utilised effectively for data augmentation purposes in healthcare domain. This method has displayed potential in enhancing classification performance and maintaining a consistent training process. To expand on this work, more research could be conducted to see how the K-CGAN method applies to other type datasets and domains. Additionally, exploring further optimization techniques could improve the model's performance. The current approach is based on backpropagation and gradient descent, which may not always be the most efficient approach for particular datasets. Through these experiments, researchers may gain a better understanding of how and when to apply this method in various applications.

To address the limitations, there is an ongoing effort to investigate various ways in which K-CGAN can be further improved. For instance, researchers are exploring different methods of data augmentation and normalisation to better represent complex datasets. This would include utilising techniques such as transfer learning and active learning for better utilisation of raw data sets. Research could develop new architectures that can handle large scale and complex data sets more efficiently than existing models used in K-CGAN. This could prove extremely beneficial for businesses or individuals with large-scale data sets that need to be analysed and synthesised. In conclusion, K-CGAN has great potential in providing high quality synthetic dataset from complex data sets. However, there is still a lot of room for further research and improvement to make the model even more efficient and effective. With the continued research, K-CGAN can be used to a great advantage in the coming years.

This research project provides a roadmap for further development and improvement of the K-CGAN model, which would help more businesses and individuals benefit from its use. The results obtained through this research will provide valuable insight into how best to utilise K-CGAN for data synthesis, helping to foster the growth of this important technology. Furthermore, it will also promote the use of K-CGAN as a reliable and efficient tool for businesses, allowing them to better analyse their complex datasets and pursue new opportunities.

6.1 Contributions

The research has successfully contributed to improving both data augmentation techniques and resolving class imbalance issues thus improving classification performance. Extensive experiments were conducted to validate the achieved research goals, and the study's contributions are highlighted below:

- **Custom Optimised K-CGAN Architecture**

The utilisation of CGAN is a unique aspect of the K-CGAN model. By conditioning the generator to produce fake data based on the target class labels, the K-CGAN aims to generate synthetic data that not only resembles the original data distribution but also aligns with specific class attributes. The K-CGAN employs different activation functions for the generator and discriminator networks. While the generator uses the widely-used ReLU activation function, the discriminator leverages the LeakyReLU activation function. This combination of activation functions allows the model to effectively handle credit card and breast cancer datasets and enhance convergence. The use of a combined loss function for the generator network, consisting of the trained discriminator loss and the KL divergence ensures that the generated samples closely match the original data distribution, improving the quality of synthetic data generation. The hyperparameters used in the K-CGAN, such as learning rate, dropout rate, neuron sizes, kernel initializer, and kernel regularizer, have been carefully optimized to enhance the model's performance, prevent overfitting, and ensure stable convergence. The deliberate setting of the activation function in the nodes to decrease noise's influence so that it's capable of handling noise effectively, making it robust to noisy data. By utilising the glorot_uniform kernel initializer, the K-CGAN addresses the exploding gradient problem, which can hinder the training process of deep neural networks.

- **Data Augmentation Method**

In K-CGAN the generator is programmed to create samples that resemble the real data by reducing the KL Divergence between the distribution of the generated data and the distribution of the real data. This leads to improved quality of the synthetic data and enhances the performance of classification and other related tasks.

- **Optimised hyperparameter settings were obtained for other GAN-based methods**

Through our research, we conducted hyperparameter tuning to discover the optimal hyperparameters for various GAN-based methods including Vanilla CGAN, NS GAN, LS GAN, SDG GAN and WGAN. The outcomes of this study can be immensely helpful in making informed decisions while selecting the appropriate data augmentation technique. By utilising these findings, researchers and practitioners can enhance their comprehension and effectively choose the most suitable approach tailored to their specific requirements.

- **Multiple classification methods implementation and feature engineering**

We implemented and tested various popular classification methods under different conditions using both imbalanced and balanced datasets. The results are valuable for selecting classification methods in minority class detection, particularly in credit card fraud and breast cancer detection. Additionally, we demonstrated the positive impact of applying SMOTE to aid in GAN training. This contribution not only improves GAN performance but also benefits other methods in the field.

- **Deployable solution**

Demonstrated deployment of the model on the cloud using Flask REST API and AWS services. The instructions are included in the Appendix sections. The synthetic data can be used to improve the performance of machine learning models or any other purposes where synthetic data may be required. The deployment of the model demonstrates its effectiveness to generate any number of samples and integrate into existing systems for data generation tasks. K-CGAN is capable of producing large quantities of data quickly and efficiently, making it an excellent choice for a wide range of data generation tasks.

- **Effectively resolving imbalanced dataset issue**

K-CGAN addresses the issue of imbalanced data, which leads to better performance of classifiers and effective detection of minority classes. Imbalanced data is a common occurrence in production environments, especially with a scarcity of the minority class. K-CGAN can generate synthetic data to augment the existing dataset, enabling classifiers to train on more balanced datasets and increase performance. This is especially beneficial in fraud detection as it enables better accuracy and more robust classification models.

- **Credit card fraud detection**

Our research utilised a genuine dataset on credit card fraud that is commonly used. The dataset shows 284,807 transactions that took place in two days, in which there were 492 fraud cases. The dataset is imbalanced since only 0.172% of all transactions are fraudulent. The K-CGAN method was used to develop a balanced credit card dataset through the generation of a synthetic dataset. This resulted in a significant enhancement of the classifiers' performance, leading to an effective approach to detect credit card fraud detection.

- **Breast cancer detection**

By subjecting our model to a different dataset, we've evaluated its ability to generalize and learn from new data. This study aims to shed light on the performance of our K-CGAN model with KL as a custom loss and its potential applications in other domains. To assess the effectiveness of our custom K-CGAN method, we conducted several experiments including model optimization using the additional breast cancer dataset. The results indicate that the proposed model performs well, as evidenced by the high F1 scores achieved on the breast cancer dataset.

- **Adaptability and scalability of the model**

The K-CGAN was developed for credit card fraud data and further tested and optimized for the Wisconsin Breast Cancer (Diagnostic) data. As a result, it produces a synthetic dataset of high quality and enhances the performance of classifiers. As such, the K-CGAN model can be easily adapted to various datasets and generate high quality synthetic data that is suitable for a wide range of tasks. This demonstrates the adaptability of the K-CGAN method as it can be optimised for different tasks and datasets. Furthermore, by using an efficient optimization algorithm, the K-CGAN can produce synthetic data quickly and efficiently while providing high accuracy results.

- **Training stability**

Using KL divergence in the loss function improves the stability of GAN training by preventing the generator from producing low-quality samples and diverging. KL divergence is a more reliable metric than other distance measurements like the Wasserstein distance or the Euclidean distance. KL divergence encourages the generator to produce samples that are more similar to the real data. This makes it easier for the discriminator to accurately identify fake samples. In addition, it helps ensure that the generated samples remain consistent over training iterations, making it easier for a model to converge and learn from its mistakes. Finally, using KL divergence in the loss function helps to increase the diversity of generated samples, making them more realistic.

- **Solution to mode collapse**

In GANs, mode collapse occurs when the generator produces a limited number of highly similar samples, which leads to a lack of diversity in the generated samples. To address this issue, a solution is to incorporate KL divergence in the loss function and the set of optimised hyperparameters, which incentivizes the generator to produce a wider range of samples that accurately represent the entire true data distribution. The algorithm is also highly adaptive and can be used to generate data with varying levels of complexity. Additionally, it has been shown to produce high-quality results when compared to other generative methods.

- **Extensive application of visualisation of synthetic and original data**

We extensively applied various visualisation techniques to both synthetic and original data during our experimentation. These techniques allowed us to gain a deeper understanding of the patterns and relationships within the dataset. To assess data quality, we used various methods including cosine similarity, bivariate and univariate correlations. These approaches provided quantitative metrics and valuable insights into the data's characteristics and distribution. Our comprehensive analysis also highlighted the superior performance of our K-CGAN method compared to existing approaches in terms of data quality and accuracy. Through this in-depth exploration, we unlocked the true potential and value of our approach, paving the way for future advancements in the field.

6.2 Further work

The K-CGAN model has undergone testing in the domains of credit card fraud and breast cancer data. It could be a promising area of research to investigate additional fields in order to further expand the model. For example, it could be beneficial to explore the effectiveness of K-CGAN in handling large datasets with high-dimensional features and complex data distributions. Additionally, exploring different methods for finding the optimal parameters for the model can help ensure that it works optimally on a wide range of datasets. Finally, researching how to improve the generative capabilities of K-CGAN to generate synthetic data that better matches the target data distribution could potentially lead to more accurate models and improved generalisation performance.

K-CGAN is a promising method for unsupervised learning in highly imbalanced datasets, and further research can help build upon its already impressive capabilities. By exploring additional fields of application and further improving the model, we can continue to unlock the potential of K-CGAN and advance unsupervised learning on imbalanced datasets. With continued research, K-CGAN could be a powerful tool for tackling challenging data problems in many areas.

Utilising K-CGAN in other data types (such as images or audio) as well as investigating different techniques for training the model, such as transfer learning and meta-learning, could also prove beneficial in improving its performance. Additionally, exploring how to use K-CGAN in combination with other models, such as deep neural networks or reinforcement learning agents, could open up exciting new possibilities for data analysis.

Finally, developing methods for validating the performance of K-CGAN could help ensure that it is being used correctly and help to improve data analysis tasks. Further exploring the potential applications, features, and limitations of K-CGAN continue to advance its development as a tool for unsupervised learning on highly imbalanced datasets.

List of References

- Asha, R.B. and KR, S.K., 2021. Credit card fraud detection using artificial neural network, *Global Transitions Proceedings*, Elsevier, Vol. 2 No. 1, pp. 35–41.
- Alejo, R., García, V., Marqués, A.I., Sánchez, J.S. and Antonio-Velázquez, J.A., 2013. Making accurate credit risk predictions with cost-sensitive mlp neural networks, *Management Intelligent Systems*, Springer, pp. 1–8.
- Ahmad, H., Kasasbeh, B., Aldabaybah, B. and Rawashdeh, E., 2023. Class balancing framework for credit card fraud detection based on clustering and similarity-based selection (SBS), *International Journal of Information Technology*, Springer, Vol. 15 No. 1, pp. 325–333.
- Abbasimehr, H. and Paki, R., 2022. Improving time series forecasting using LSTM and attention pARKmodels, *Journal of Ambient Intelligence and Humanized Computing*, Springer, pp. 1–19.
- Abbasimehr, H., Shabani, M. and Yousefi, M., 2020. An optimized model using LSTM network for demand forecasting, *Computers & Industrial Engineering*, Elsevier, Vol. 143, p. 106435.
- Abdulhayan, S., Firdouse, L., Haleema, N. and Anfeeda, Z., 2023. Credit Card Fraud Detection Using Machine Learning, *Recent Trends in Information Technology and Its Application*, Vol. 6 No. 2, pp. 1–4.
- Alharbi, A., Alshammari, M., Okon, O.D., Alabrah, A., Rauf, H.T., Alyami, H. and Meraj, T., 2022. A novel text2IMG mechanism of credit card fraud detection: a deep learning approach, *Electronics*, MDPI, Vol. 11 No. 5, p. 756.
- Ashwin, V., Menon, V., Devagopal, A.M., Nived, P.A. and Udayan Divya, J., 2023. Detection of Fraudulent Credit Card Transactions in Real Time Using SparkML and Kafka, *Proceedings of 3rd International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications: ICMISC 2022*, Springer, pp. 285–295.
- Arjovsky, M., Chintala, S. and Bottou, L., 2017. Wasserstein GAN, In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 70, pp. 214–223.
- Alejo, R., García, V., Marqués, A.I., Sánchez, J.S. and Antonio-Velázquez, J.A., 2013. Making accurate credit risk predictions with cost-sensitive mlp neural networks, *Management Intelligent Systems*, Springer, pp. 1–8.
- Abdi, L. and Hashemi, S., 2015. To combat multi-class imbalanced problems by means of over-sampling techniques, *IEEE Transactions on Knowledge and Data Engineering*, IEEE, Vol. 28 No. 1, pp. 238–251.
- Abdallah, A., Maarof, M.A. and Zainal, A., 2016. Fraud detection system: A survey, *Journal of Network and Computer Applications*, Elsevier, Vol. 68, pp. 90–113.
- Awoyemi, J. O., Adetunmbi, A. O. and Oluwadare, S. A., 2017. Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI)*, 1–9.
- A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah and A. Hussain, Comparing Oversampling Techniques to handle the class imbalance problem: a customer churn prediction case study, in *IEEEAccess*, pages 7940-7954, 2016.

- Ahmad, A. and Brown, G., 2013. Random projection random discretization ensembles—Ensembles of linear multivariate decision trees, *IEEE Transactions on Knowledge and data Engineering*, 26(5), pp. 1225–1239.
- Arjovsky, M. and Bottou, L., 2017. Towards Principled Methods for Training Generative Adversarial Networks, 1–17. Available at: <https://doi.org/10.48550/arXiv.1701.04862>.
- Anh, N. T. N., Khanh, T. Q., Dat, N. Q., Amouroux, E., and Solanki, V. K., 2020. Fraud detection via deep neural variational autoencoder oblique random forest. 2020 IEEE-HYDCON, 1–6.
- Alarsan, F. I. and Younes, M., 2021. Best Selection of Generative Adversarial Networks Hyper-Parameters Using Genetic Algorithm. *SN Computer Science* [online], 2 (4). Available from: <http://dx.doi.org/10.1007/s42979-021-00689-3>.
- Arora, S., Ge, R., Neyshabur, B., and Zhang, Y., 2017. Stronger generalization bounds for deep nets via a compression approach. In the International Conference on Learning Representations (ICLR).
- A. D. Trister, D. S. M. Buist, and C. I. Lee, “Will Machine Learning Tip the Balance in Breast Cancer Screening?,” *JAMA Oncol*, vol. 3, no. 11, p. 1463, Nov. 2017, doi: 10.1001/jamaoncol.2017.0473.
- Bishop, C. M., 2006. *Pattern Recognition and Machine Learning*. Springer. Microsoft Research Cambridge, Cambridge, United Kingdom.
- Breskuvienė, D. and Dzemyda, G., 2023. Categorical Feature Encoding Techniques for Improved Classifier Performance when Dealing with Imbalanced Data of Fraudulent Transactions, *International Journal of Computers Communications and Control*, Vol. 18 No. 3.
- Bockel-Rickermann, C., Verdonck, T. and Verbeke, W., 2022. Fraud Analytics: A Decade of Research--Organizing Challenges and Solutions in the Field, *ArXiv Preprint ArXiv:2212.04329*.
- Brownlee, J., 2019. How to Calculate the KL Divergence for Machine Learning - MachineLearningMastery.com
- Büttcher, S., Clarke, L.A. C., and Cormack, G. V., 2010. *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press.
- Bhagyashree, Kushwaha, V. and Nandi, G. C., 2020. Study of Prevention of Mode Collapse in Generative Adversarial Network (GAN). 2020 IEEE 4th Conference on Information & Communication Technology (CICT) [online]. Available from: <http://dx.doi.org/10.1109/cict51604.2020.9312049>.
- Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning (ICML)* (Vol. 28, No. 3).
- Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., and Cox, D. D., 2015. Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms. *Proceedings of the 12th Python in Science Conference*, 13-20.
- Bergstra, J., and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281-305.
- Ba, H., 2019. Improving detection of credit card fraudulent transactions using generative adversarial networks. *ArXiv Preprint ArXiv:1907.03355*.
- Breiman, L., 1996. Bagging predictors, *Machine Learning*, 24(2), pp. 123–140. Available at: <https://doi.org/10.1007/bf00058655>.

- Batista, G.E., Prati, R.C. and Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data, *ACM SIGKDD Explorations Newsletter*, ACM New York, NY, USA, Vol. 6 No. 1, pp. 20–29.
- Breiman, L., 2001. Random Forests, *Machine Learning*, 45(1), pp. 5–32. doi:10.1023/a:1010933404324.
- Bansal, M.A., Sharma, D.R. and Kathuria, D.M., 2021. A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications. *ACM Computing Surveys (CSUR)*.
- Bahnsen, A.C., Aouada, D., Stojanovic, A. and Ottersten, B., 2016. Feature engineering strategies for credit card fraud detection, *Expert Systems with Applications*, Elsevier, Vol. 51, pp. 134–142.
- C. -T. Peng, Y. -K. Chan and S. S. Yu, "Data Imbalance in Immunity Bone Age Assessment System Using Independent Autoencoders," 2022 IEEE 4th Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), Tainan, Taiwan, 2022, pp. 156-158, doi: 10.1109/ECBIOS54627.2022.9944988.
- Chiaroni, F., Rahal, M.-C., Hueber, N. and Dufaux, F., 2019. Hallucinating A Cleanly Labeled Augmented Dataset from A Noisy Labeled Dataset Using GAN. 2019 IEEE International Conference on Image Processing (ICIP) [online]. Available from: <http://dx.doi.org/10.1109/icip.2019.8803632>.
- Chandrakanth, P., 2023. A cost sensitive Random Forest Algorithm for Detecting a credit card Fraud techniques. Available at: https://ejmcm.com/article_22386_3d1179ae7834eb4033b91cc3805465d9.pdf (Accessed on: 09 June 2023)
- Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K. and Bengio, Y., 2015. Attention-based models for speech recognition, *Advances in Neural Information Processing Systems*, Vol. 28.
- Charitou, C., Dragicevic, S. and Garcez, A. d'Avila, 2021. Synthetic Data Generation for Fraud Detection using GANs, *arXiv*, pp. 1–8.
- Chawla, N. V, Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002, SMOTE: synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research*, Vol. 16, pp. 321–357.
- Chen, T. and Guestrin, C., 2016. XGBoost, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. doi:10.1145/2939672.2939785.
- Cover, T. and Hart, P., 1967. Nearest neighbor Pattern Classification, *IEEE Transactions on Information Theory*, 13(1), pp. 21–27. doi:10.1109/tit.1967.1053964.
- Chen, J.; Shen, Y.; Ali, R. Credit card fraud detection using sparse autoencoder and generative adversarial network. In *Proceedings of the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 1–3 November 2018; pp. 1054–1059.
- Chen, X.W.; Wasikowski, M. Fast: A roc-based feature selection metric for small samples and imbalanced data classification problems. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, USA, 24–27 August 2008; pp. 124–132.
- Charitou, C., Garcez, A. d'Avila and Dragicevic, S. 2020, Semi-supervised gans for fraud detection, 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1–8.
- Cordón, I.; García, S., Fernández, A. and Herrera, F., 2018. Imbalance: Oversampling algorithms for imbalanced classification in R. *Knowledge-Based Syst.* 2018, 161, 329–341.

- Cao, P., Liu, X., Zhang, J., Zhao, D., Huang, M. and Zaiane, O., 2017, ℓ_2 , 1 norm regularized multi-kernel based joint nonlinear feature selection and over-sampling for imbalanced data classification, *Neurocomputing*, Elsevier, Vol. 234, pp. 38–57.
- Caruana, R., Niculescu-Mizil A., Crew G. and Ksikes A., 2004. Ensemble selection from libraries of models, in *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pp. 137–144. Available at: <https://doi.org/10.1145/1015330.1015432>.
- Cao, Y.-J., Jia, L.-L., Chen, Y.-X., Lin, N., Yang, C., Zhang, B., Liu, Z., Li, X.-X., and Dai, H.-H., 2018. Recent advances of generative adversarial networks in computer vision. *IEEE Access* 7, 14985–15006.
- Chou, H.-P., Chang, S.-C., Pan, J.-Y., Wei, W., and Juan, D.-C., 2020. Remix: rebalanced mixup. In: *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI* 16. Springer, pp. 95–110.
- Chole, V., Mukherjee, A., Gaikwad, K., Gawai, P., Bagde, P., Mahule, R. and Pawar, P., 2022. Revelation of Credit Card Fraud using Machine Learning Algorithm.
- Chen, H., 2021. Challenges and Corresponding Solutions of Generative Adversarial Networks (GANs): A Survey Study. *Journal of Physics: Conference Series* [online], 1827 (1), 012066. Available from: <http://dx.doi.org/10.1088/1742-6596/1827/1/012066>.
- Chen, J. I.-Z., and Lai, K.-L., 2021. Deep convolution neural network model for credit-card fraud detection and alert. *Journal of Artificial Intelligence*, 3(02), 101–112.
- Dev, K., Khowaja, S.A., Bist, A.S., Saini, V. and Bhatia, S., 2021 “Triage of potential COVID-19 patients from chest X-ray images using hierarchical convolutional networks”, *Neural Computing and Applications*, Springer, pp. 1–16.
- Ding, H., Sun, Y., Wang, Z., Huang, N., Shen, Z. and Cui, X., 2023. RGAN-EL: A GAN and ensemble learning-based hybrid approach for imbalanced data classification, *Information Processing and Management*, Elsevier, Vol. 60 No. 2, p. 103235.
- Das, B., Krishnan, N.C. and Cook, D.J., 2014. RACOG and wRACOG: Two probabilistic oversampling techniques, *IEEE Transactions on Knowledge and Data Engineering*, IEEE, Vol. 27 No. 1, pp. 222–234.
- Derrac, J., Garcia, S., Sanchez, L. and Herrera, F., 2015. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Logic Soft Comput*, 17.
- Douzas, G. and Bacao, F., 2018. Effective data generation for imbalanced learning using conditional generative adversarial networks, *Expert Systems with Applications*, Elsevier, Vol. 91, pp. 464–471.
- Dietterich, T.G., 2000. Ensemble methods in machine learning, in *Lecture Notes in Computer Science* (including subseries *Lecture Notes in Artificial Intelligence* and *Lecture Notes in Bioinformatics*). Springer, pp. 1–15. Available at: https://doi.org/10.1007/3-540-45014-9_1.
- Doria, M., Luciano, E. and Semeraro, P., 2022. Machine learning techniques in joint default assessment, *ArXiv Preprint ArXiv:2205.01524*.
- Ding, Z., Jiang, S. and Zhao, J., 2022. Take a close look at mode collapse and vanishing gradient in GAN. *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)* [online]. Available from: <http://dx.doi.org/10.1109/icetci55101.2022.9832406>.

- Drew, J. H., Glen, A. G. and Leemis, L. M., 2000. Computing the cumulative distribution function of the Kolmogorov–Smirnov statistic. *Computational Statistics & Data Analysis* [online], 34 (1), 1–15. Available from: [http://dx.doi.org/10.1016/s0167-9473\(99\)00069-9](http://dx.doi.org/10.1016/s0167-9473(99)00069-9).
- Dubey, A. K., and Jain, V., 2019. Comparative study of convolution neural network's relu and leaky-relu activation functions. *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*, 873–880.
- Duchi, J., Hazan, E., and Singer, Y., 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2121-2159.
- Doria, M., Luciano, E., and Semeraro, P., 2022. Machine learning techniques in joint default assessment. *arXiv Prepr. arXiv2205.01524*.
- DeMaris, A., 1995. A tutorial in logistic regression. *Journal of Marriage and the Family*, pp.956-968.
- Esmail, F.S., Alsheref, F.K. and Aboutabl, A.E., 2023. Review of Loan Fraud Detection Process in the Banking Sector Using Data Mining Techniques, *International Journal of Electrical and Computer Engineering Systems*, Elektrotehnički fakultet Sveučilišta JJ Strossmayera u Osijeku, Vol. 14 No. 2, pp. 229–239.
- Elyan, E., Moreno-Garcia, C.F. and Jayne, C., 2021. CDSMOTE: class decomposition and synthetic minority class oversampling technique for imbalanced-data classification, *Neural Computing and Applications*, Springer, Vol. 33 No. 7, pp. 2839–2851.
- Efimov, D., Xu, D., Kong, L., Nefedov, A. and Anandakrishnan, A., 2020. Using generative adversarial networks to synthesize artificial financial datasets.
- Eckerli, F. and Osterrieder, J., 2021. Generative Adversarial Networks in finance: an overview, 1–22.
- Fanai, H. and Abbasimehr, H., 2023. A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection, *Expert Systems with Applications*, Elsevier, p. 119562.
- Fu, S., Yu, X., Tian, Y., 2022. Cost sensitive v-support vector machine with LINEX loss. *Inf. Process. Manag.* 59, 102809.
- Fan, X., Guo, X., Chen, Q., Chen, Y., Wang, T. and Zhang, Y., 2022, Data augmentation of credit default swap transactions based on a sequence GAN, *Information Processing & Management*, Elsevier, Vol. 59 No. 3, p. 102889.
- Fan, Q., Wang, Z. and Gao, D., 2016. One-sided dynamic undersampling no-propagation neural networks for imbalance problem. *Engineering Applications of Artificial Intelligence*, 53, 62–73.
- Fiore, U., De Santis, A., Perla, F., Zanetti, P. and Palmieri, F., 2019. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection, *Information Sciences*, Elsevier, Vol. 479, pp. 448–455.
- Fernández-Delgado, M., Cernadas E. and Barro S., 2014. Do we need hundreds of classifiers to solve real world classification problems?, *Journal of Machine Learning Research*, 15(1), pp. 3133–3181.
- Figueira, A. and Vaz, B., 2022. Survey on Synthetic Data Generation, Evaluation Methods and GANs. *Mathematics* [online], 10 (15), 2733. Available from: <http://dx.doi.org/10.3390/math10152733>.
- Faraji, Z. (2022). A Review of Machine Learning Applications for Credit Card Fraud Detection with A Case study. *SEISENSE Journal of Management*, 5(1), 49–59.

- Goodfellow, I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014. Generative Adversarial Networks, *Communications of the ACM*, 63(11), pp. 139–144. doi:10.1145/3422622.
- Goodfellow, I., Bengio, Y., and Courville, A., 2016. *Deep Learning*. MIT Press. Chapter 6: Deep Feedforward Networks, pp. 162-166.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., and Bengio, Y., 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems (NIPS)*, 2672-2680.
- Gatys, L.A., Ecker, A.S. and Bethge, M., 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2414-2423).
- Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. and Herrera, F., 2011. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, Vol. 42 No. 4, pp. 463–484.
- Ger, S. and Klabjan, D., 2019. Autoencoders and generative adversarial networks for anomaly detection for sequences, *ArXiv Preprint ArXiv:1901.02514*, Sep.
- Gupta, G.K. and Sharma, D.K., 2022. A review of overfitting solutions in smart depression detection models, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, pp. 145–151.
- Grandini, M., Bagli, E. and Visani, G., 2020. Metrics for multi-class classification: An overview. *arXiv 2020*, arXiv:2008.05756.
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A.C., 2017. Improved training of wasserstein gans, *Advances in Neural Information Processing Systems*, Vol. 30.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2020. Generative adversarial networks. *Communications of the ACM*, 63(11), 139–144.
- Günder, M., Piatkowski, N., Bauckhage, C., 2022. Full Kullback-Leibler-Divergence Loss for Hyperparameter-free Label Distribution Learning. *ArXiv Preprint ArXiv:2209.02055*.
- Galdran, A., Carneiro, G., Ballester, M.A.G., 2023. On the Optimal Combination of Cross-Entropy and Soft Dice Losses for Lesion Segmentation with Out-of-Distribution Robustness. In: *Diabetic Foot Ulcers Grand Challenge: Third Challenge, DFUC 2022, Held in Conjunction with MICCAI 2022, Singapore, September 22, 2022, Proceedings*. Springer, pp. 40–51.
- Gangwar, A. K., and Ravi, V., 2019. Wip: Generative adversarial network for oversampling data in credit card fraud detection. *International Conference on Information Systems Security*, 123–134.
- Glorot, X., and Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)* (pp. 249-256).
- H. He, E.A. Garcia, Learning from imbalanced data, *IEEE Trans. Knowl. Data Eng.* 21 (9) (2009) 1263-1284, 2009.
- He, H., Bai Y., Garcia A. E., Li S., 2008. Adasyn: Adaptive Synthetic Sampling Approach for imbalanced learning, *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328. doi:10.1109/ijcnn.2008.4633969.

- Han, H., Wang, W.-Y. and Mao, B.-H., 2005. Borderline-smote: A new over-sampling method in imbalanced data sets learning, *Advances in Intelligent Computing: International Conference on Intelligent Computing, ICIC 2005, Hefei, China*, pp. 878–887. doi:10.1007/11538059_91.
- Hwang, J. and Kim, K., 2020. An Efficient Domain-Adaptation Method using GAN for Fraud Detection. *International Journal of Advanced Computer Science and Applications*, 11 (11), 1–11.
- Hilal W., Gadsden S.A., and Yawney J., 2022. Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advance. *Expert Systems with Applications*, 193.
- Hajek, P., Abedin, M.Z., and Sivarajah, U., 2022. Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework. *Inf. Syst. Front.* 1–19.
- Hong, Y., Hwang, U., Yoo, J. and Yoon, S., 2019. How Generative Adversarial Networks and Their Variants Work. *ACM Computing Surveys [online]*, 52 (1), 1–43. Available from: <http://dx.doi.org/10.1145/3301282>.
- He, H., and Wu, D., 2019. Imbalanced Learning: Foundations, Algorithms, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9), 3123-3140.
- H. Chou, S. Chang, J. Pan, W. Wei, D. Juan, Remix: Rebalanced Mixup, in arXiv, pages 1-18, 2020.
- Hordri, N.F., Yuhaniz, S.S., Azmi, N.F.M. and Shamsuddin, S.M., 2018. Handling class imbalance in credit card fraud using resampling methods, *Int. J. Adv. Comput. Sci. Appl*, Vol. 9 No. 11, pp. 390–396.
- Ioffe, S., and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning (ICML)*, 37, 448-456.
- Imbalanced-learn.org. (n.d.). SMOTE — Version 0.9.0. [online] Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (Accessed 23 June 2023)
- Imbalanced-learn.org. (n.d.). ADASYN — Version 0.9.1. [online] Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html (Accessed 23 June 2023)
- Imbalanced-learn.org. (n.d.). BorderlineSMOTE — Version 0.10.1. [online] Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html (Accessed 24 June 2023) (Accessed 23 June 2023).
- I. Ullah and Q. H. Mahmoud, A Framework for Anomaly Detection in IoT Networks Using Conditional Generative Adversarial Networks, in *IEEE Access*, vol. 9, pp. 165907-165931, 2021, doi:
- Jain, R., Gour, B., and Dubey, S., 2016. A hybrid approach for credit card fraud detection using rough set and decision tree technique. *International Journal of Computer Applications*, 139(10), 1–6.
- Jiang, C., Song, J., Liu, G., Zheng, L., and Luan, W., 2018. Credit card fraud detection: A novel approach using aggregation strategy and feedback mechanism. *IEEE Internet of Things Journal*, 5(5), 3637–3647.
- Karthika, J. and Senthilselvi, A., 2023. An integration of deep learning model with Navo Minority Over-Sampling Technique to detect the frauds in credit cards, *Multimedia Tools and Applications*, Springer, pp. 1–18.
- Kumar, Y. and Gupta, S. (2023), “Effectiveness of Machine and Deep Learning for Blockchain Technology in Fraud Detection and Prevention”, *Applications of Artificial Intelligence, Big Data and Internet of Things in Sustainable Development*, CRC Press, pp. 287–307.

Kumar, P.; Iqbal, F. Credit card fraud identification using machine learning approaches. In Proceedings of the 2019 1st International conference on innovations in information and communication technology (ICIICT), Chennai, India, 25–26 April 2019; pp. 1–4.

Karthik, V.S.S., Mishra, A. and Reddy, U.S. (2022), “Credit card fraud detection by modelling behaviour pattern using hybrid ensemble model”, *Arabian Journal for Science and Engineering*, Springer, pp. 1–11.

Kaur, P. and Gosain, A., 2018. Comparing the behavior of oversampling and undersampling approach of class imbalance learning by combining class imbalance problem with noise, *ICT Based Innovations: Proceedings of CSI 2015*, Springer, pp. 23–30.

Kar, N., Saha, A., and Deb, S., 2020. Trends in Computational Intelligence, Security and Internet of Things: Third International Conference, ICCISIoT 2020, Tripura, India, December 29-30, 2020, Proceedings, Vol. 1358, Springer Nature.

Kullback, S. and Leibler, R.A., 1951. On information and sufficiency, *The Annals of Mathematical Statistics*, JSTOR, Vol. 22 No. 1, pp. 79–86.

Kaggle (2021). Kaggle credit card fraud detection. Available at: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>. (Accessed on: 16 June 2023).

Kingma, D. P., and Ba, J., 2014. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

Kossale, Y., Airaj, M., and Darouichi, A., 2022. Mode Collapse in Generative Adversarial Networks: An Overview. 2022 8th International Conference on Optimization and Applications (ICOA), 1–6.

Kim, T., Oh, J., Kim, N., Cho, S., and Yun, S.-Y., 2021. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. ArXiv Preprint ArXiv:2105.08919.

Luong, M.-T., Pham, H. and Manning, C.D., 2015. Effective approaches to attention-based neural machine translation, ArXiv Preprint ArXiv:1508.04025.

Langevin, A., Cody, T., Adams, S. and Beling, P., 2021. Synthetic data augmentation of imbalanced datasets with generative adversarial networks under varying distributional assumptions: A case study in credit card fraud detection. *Journal of the Operational Research Society*, pp.1-28.

Langevin, A., Cody, T., Adams, S. and Beling, P., 2022. Generative adversarial networks for data augmentation and transfer in credit card fraud detection, *Journal of the Operational Research Society*, Taylor & Francis, Vol. 73 No. 1, pp. 153–180.

Lamba, H. Credit Card Fraud Detection in Real-Time. Ph.D. Thesis, California State University San Marcos, San Marcos, CA, USA, 2020.

López, V., Fernández, A., García, S., Palade, V. and Herrera, F., 2013. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, *Information Sciences*, Elsevier, Vol. 250, pp. 113–141.

Lim, P., Goh, C.K. and Tan, K.C., 2016. Evolutionary cluster-based synthetic oversampling ensemble (eco-ensemble) for imbalance learning, *IEEE Transactions on Cybernetics*, IEEE, Vol. 47 No. 9, pp. 2850–2861.

LeCun, Y., Bengio, Y., and Hinton, G., 2015. Deep learning. *Nature*, 521(7553), 436–444.

Lei, J. Z., and Ghorbani, A. A., 2012. Improved competitive learning neural networks for network intrusion and fraud detection. *Neurocomputing*, 75(1), 135–145.

- Luo, H., Men, Y., and Zheng, Q., 2022. Sparse Autoencoders with KL Divergence.
- Li, K. and Kang, D.-K., 2022. Enhanced Generative Adversarial Networks with Restart Learning Rate in Discriminator, *Applied Sciences*, 12(3), p. 1191. doi:10.3390/app12031191.
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A., 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1), 6765–6816.
- Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O., 2018. Are GANs Created Equal? A Large-Scale Study. arXiv preprint arXiv:1711.10337.
- Li J., Zhu Q., Wu Q. and Fan Z., 2021. A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbours. *Inf. Sci.* 2021, 565, 438–455.
- M. M. Saripuddin, A. Suliman and S. S. Sameon, "Impact of Resampling and Deep Learning to Detect Anomaly in Imbalance Time-Series Data," 2022 14th International Conference on Computer Research and Development (ICCRD), Shenzhen, China, 2022, pp. 37–41, doi: 10.1109/ICCRD54409.2022.9730424.
- M. Kowal, P. Filipczuk, A. Obuchowicz, J. Korbicz, and R. Monczak, "Computer-aided diagnosis of breast cancer based on fine needle biopsy microscopic images," *Comput Biol Med*, vol. 43, no. 10, pp. 1563–1572, Oct. 2013, doi: 10.1016/j.combiomed.2013.08.003.
- Mirza, M. and Osindero, S. (2014) 'Conditional Generative Adversarial Nets', arXiv , pp. 1–7.
- Mao, X., Li Q., Xie H., Lau Y.K. R., Wang Z., Smolley S. P., 2017. Least squares generative adversarial networks, 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2813–2821. doi:10.1109/iccv.2017.304.
- Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M.-S., Zeineddine, H., 2019. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access* 7, 93010–93022.
- Maldonado, S., Vairetti, C., Fernandez, A. and Herrera, F., 2022. FW-SMOTE: A feature-weighted oversampling approach for imbalanced classification, *Pattern Recognition*, Elsevier, Vol. 124, p. 108511.
- Mohindru, G., Mondal, K., and Banka, H., 2021. Different hybrid machine intelligence techniques for handling IoT-based imbalanced data. *CAAI Transactions on Intelligence Technology*, 6(4), 405–416.
- Mqadi, N., Naicker, N. and Adeliyi, T., 2021. A SMOTE based oversampling data-point approach to solving the credit card data imbalance problem in financial fraud detection, *International Journal of Computing and Digital Systems*, University of Bahrain, Vol. 10 No. 1, pp. 277–286.
- Manjurul Ahsan, M., Shahin Ali, M. and Siddique, Z., 2022. Imbalanced Class Data Performance Evaluation and Improvement using Novel Generative Adversarial Network-based Approach: SSG and GBO, ArXiv E-Prints, p. arXiv-2210.
- Moniz, N., Branco, P. and Torgo, L., 2017. Evaluation of ensemble methods in imbalanced regression tasks', in *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*. PMLR, pp. 129–140.
- Mescheder, L., Geiger, A. and Nowozin, S., 2018. Which Training Methods for GANs do actually Converge?, 1–10.
- McIver, S., 2021. Can Generative Adversarial Networks Help Us Fight Financial Fraud? , Technological University Dublin.

- Mullick, S.S., Datta, S., and Das, S., 2019. Generative adversarial minority oversampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1695–1704.
- Maas, A. L., Hannun, A. Y., and Ng, A. Y., 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In Proceedings of the 30th International Conference on Machine Learning (ICML-13) (Vol. 28, No. 3).
- Mescheder, L., Nowozin, S., and Geiger, A., 2018. Which training methods for GANs do actually converge? In International Conference on Machine Learning (ICML) (Vol. 80, pp. 3478-3487).
- Masters, D., and Luschi, C., 2018. Revisiting Small Batch Training for Deep Neural Networks. arXiv preprint arXiv:1804.07612.
- Ni, L., Li, J., Xu, H., Wang, X. and Zhang, J., 2023. Fraud Feature Boosting Mechanism and Spiral Oversampling Balancing Technique for Credit Card Fraud Detection, IEEE Transactions on Computational Social Systems, IEEE.
- Ngwenduna, K.S.; Mbuva, R. Alleviating class imbalance in actuarial applications using generative adversarial networks. Risks 2021, 9, 49.
- Nabulsi, Z., Sellergren, A., Jamshy, S., Lau, C., Santos, E., Kiraly, A.P., Ye, W., 2021. Deep learning for distinguishing normal versus abnormal chest radiographs and generalization to two unseen diseases tuberculosis and COVID-19, Scientific Reports, Nature Publishing Group, Vol. 11 No. 1, pp. 1–15.
- Ng, W.W.Y., Xu, S., Zhang, J., Tian, X., Rong, T. and Kwong, S., 2020. Hashing-based undersampling ensemble for imbalanced pattern classification problems, IEEE Transactions on Cybernetics, IEEE, Vol. 52 No. 2, pp. 1269–1279.
- Nair, V., and Hinton, G. E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML-10) (pp. 807-814).
- Ngo, P. C., Winarto, A. A., Kou, C. K. L., Park, S., Akram, F., and Lee, H. K., 2019. Fence GAN: Towards better anomaly detection. 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 141–148.
- Nezamzadeh-Ejeh, S., and Sadeghkhani, I., 2020. HIF detection in distribution networks based on Kullback–Leibler divergence. IET Generation, Transmission & Distribution, 14(1), 29–36.
- Olszewski, D., 2012b. Employing Kullback-Leibler divergence and Latent Dirichlet Allocation for fraud detection in telecommunications, Intelligent Data Analysis, IOS Press, Vol. 16 No. 3, pp. 467–485.
- Olszewski, D. 2012a. A probabilistic approach to fraud detection in telecommunications, Knowledge-Based Systems, Elsevier, Vol. 26, pp. 246–258.
- Prusti, D., Rout, J.K. and Rath, S.K., 2023. Detection of credit card fraud by applying genetic algorithm and particle swarm optimization, Machine Learning, Image Processing, Network Security and Data Sciences: Select Proceedings of 3rd International Conference on MIND 2021, Springer, pp. 357–369.
- Paasch, C.A. Credit Card Fraud Detection Using Artificial Neural Networks Tuned by Genetic Algorithms; Hong Kong University of Science and Technology: Hong Kong, China, 2008.
- Prusti, D.; Rath, S.K. Web service based credit card fraud detection by applying machine learning techniques. In Proceedings of the TENCON 2019-2019 IEEE Region 10 Conference (TENCON), Kochi, India, 17–20 October 2019; pp. 492–497.

- Phua, C., Lee, V., Smith, K. and Gayler, R., 2010. A comprehensive survey of data mining-based fraud detection research, ArXiv Preprint ArXiv:1009.6119.
- Phua, C., Alahakoon, D., and Lee, V., 2004. Minority report in fraud detection: classification of skewed data. *Acm Sigkdd Explorations Newsletter*, 6(1), 50–59.
- Prati, R.C., Batista, G.E. and Monard, M.C., 2004. Learning with class skews and small disjuncts, *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence*, Sao Luis, Maranhao, Brazil, September 29-October 1, 2004. *Proceedings 17*, Springer, pp. 296–306.
- Perera, P., Nallapati, R. and Xiang, B., 2019. OCGAN: One-class Novelty Detection Using GANs with Constrained Latent Representations.
- Padmanabhuni, S. S., and Gera, P., 2022. Synthetic data augmentation of tomato plant leaf using meta intelligent generative adversarial network: milgan. *International Journal of Advanced Computer Science and Applications*, 13(6).
- Pitsane, M.Y., Mogale, H., van Rensburg, J.T.J., 2022. Improving Accuracy of Credit Card Fraud Detection Using Supervised Machine Learning Models and Dimension Reduction. In: *International Conference on Intelligent and Innovative Computing Applications*. pp. 290–301.
- Park, N., Gu, Y. H., and Yoo, S. J., 2021. Synthesising Individual Consumers' Credit Historical Data Using Generative Adversarial Networks. *Applied Sciences*, 11(3), 1126.
- Pfenninger, M., Rikli, S. and Bigler, D. N., 2021. Wasserstein GAN: Deep Generation Applied on Financial Time Series. *SSRN Electronic Journal* [online]. Available from: <http://dx.doi.org/10.2139/ssrn.3877960>.
- Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F. and Zheng, Y., 2019. Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access* [online], 7, 36322–36333. Available from: <http://dx.doi.org/10.1109/access.2019.2905015>.
- Pavan K., M. R. and Jayagopal, P., 2021. Multi-class imbalanced image classification using conditioned GANs. *International Journal of Multimedia Information Retrieval* [online], 10 (3), 143–153. Available from: <http://dx.doi.org/10.1007/s13735-021-00213-6>.
- Panigrahi, S., Kundu, A., Sural, S., and Majumdar, A. K., 2009. Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning. *Information Fusion*, 10(4), 354–363.
- Prokhorov, V., Shareghi, E., Li, Y., Pilehvar, M. T., and Collier, N., 2019. On the importance of the Kullback-Leibler divergence term in variational autoencoders for text generation. *ArXiv Preprint ArXiv:1909.13668*.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986. Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536.
- Radford, A., Metz, L., and Chintala, S., 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Rtayli, N. 2022, An Efficient Deep Learning Classification Model for Predicting Credit Card Fraud on Skewed Data, *Journal of Information Security and Cybercrimes Research*, Vol. 5 No. 1, pp. 61–75.
- Rosenblatt, F., 1957. The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

- Ratner, A.J., Ehrenberg, H., Hussain, Z., Dunnmon, J. and Ré, C., 2017. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 30.
- Ramik Rawal, BREAST CANCER PREDICTION USING MACHINE LEARNING, *JETIR*, vol. 7, no. 5, pp. 13–24, May 2020.
- Smith, L. N., and Topin, N., 2020. Super-convergence: Very fast training of residual networks using large learning rates. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Sauber-Cole, R. and Khoshgoftaar, T. M., 2022. The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey. *Journal of Big Data*, 9 (1), 98.
- Sarah, S., Singh, V., Gourisaria, M. K., and Singh, P. K., 2021. Retinal Disease Detection using CNN through Optical Coherence Tomography Images. 2021 5th International Conference on Information Systems and Computer Networks (ISCON), 1–7.
- Saxena, D. , Cao, J., 2021. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *ACM Computing Surveys [online]*, 54 (3), 1–42. Available from: <http://dx.doi.org/10.1145/3446374>.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1), 1929-1958.
- Smith, J. A., 2020. *A Comprehensive Guide to Data Preprocessing in Machine Learning: Methods and Techniques*.
- Srivastava, A., Kundu, A., Sural, S., and Majumdar, A., 2008. Credit card fraud detection using hidden Markov model. *IEEE Transactions on Dependable and Secure Computing*, 5(1), 37–48.
- Shlens, J., 2014. Notes on kullback-leibler divergence and likelihood. *arXiv Prepr. arXiv1404.2000*.
- Shen, J., 2021. Credit card fraud detection using autoencoder-based deep neural networks, 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), IEEE, pp. 673–677.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X., 2016. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 29.
- Sabuhi, M., Zhou, M., Bezemer, C.-P. and Musilek, P., 2021. Applications of Generative Adversarial Networks in Anomaly Detection: A Systematic Literature Review.
- Singh, A., Ranjan, R.K. and Tiwari, A., 2021. Credit card fraud detection under extreme imbalanced data: a comparative study of data-level algorithms, *Journal of Experimental & Theoretical Artificial Intelligence*, Taylor & Francis, pp. 1–28.
- S. Shams, R. Platania, J. Zhang, J. Kim, K. Lee, and S.-J. Park, “Deep Generative Breast Cancer Screening and Diagnosis,” 2018, pp. 859–867. doi: 10.1007/978-3-030-00934-2_95.
- Sun, J., Li, H., Fujita, H., Fu, B. and Ai, W., 2020., Class-imbalanced dynamic financial distress prediction based on Adaboost-SVM ensemble combined with SMOTE and time weighting, *Information Fusion*, Elsevier, Vol. 54, pp. 128–144.
- Singh, A., Ranjan, R.K., Tiwari, A., 2022. Credit card fraud detection under extreme imbalanced data: a comparative study of data-level algorithms. *J. Exp. Theor. Artif. Intell.* 34, 571–598.

- Sethia, A., Patel, R. and Raut, P. (2018). Data augmentation using generative models for credit card fraud detection. 2018 4th International Conference on Computing Communication and Automation (ICCCA), 1–6.
- Shorten C. and Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1), pp.1-48.
- Shannon, M., Poole, B., Mariooryad, S., Bagby, T., Battenberg, E., Kao, D., Stanton, D. and Skerry-Ryan, R.J., 2020. Non-saturating GAN training as divergence minimization. arXiv preprint arXiv:2010.08029.
- Sultana, J., Usha Rani, M. and Farquad, M.A.H., 2020. An extensive survey on some deep-learning applications, *Emerging Research in Data Engineering Systems and Computer Communications: Proceedings of CCODE 2019*, Springer, pp. 511–519.
- Salekshahrezaee, Z., Leevy, J.L. and Khoshgoftaar, T.M., 2023. The effect of feature extraction and data sampling on credit card fraud detection, *Journal of Big Data*, SpringerOpen, Vol. 10 No. 1, pp. 1–17.
- Sanober, S., Alam, I., Pande, S., Arslan, F., Rane, K.P., Singh, B.K., Khamparia, A., 2021. An enhanced secure deep learning algorithm for fraud detection in wireless communication, *Wireless Communications and Mobile Computing*, Hindawi Limited, Vol. 2021, pp. 1–14.
- Saia, R. and Carta, S., 2019. Evaluating the benefits of using proactive transformed-domain-based techniques in fraud detection tasks., *Future Generation Computer Systems*, Elsevier, Vol. 93, pp. 18–32.
- Soh, W.W., Yusuf, R.M., 2019. Predicting credit card fraud on an imbalanced data. *Int. J. Data Sci. Adv. Anal.* (ISSN 2563-4429) 1, 12–17.
- Sohony, I., Pratap, R. and Nambiar, U., 2018. Ensemble learning for credit card fraud detection, *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pp. 289–294.
- Scikit-learn.org. (n.d.). `sklearn.linear_model.LogisticRegression` — scikit-learn 0.21.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (Accessed 23 June 2023)
- Scikit-learn.org. (n.d.). `sklearn.neural_network.MLPClassifier` — scikit-learn 0.24.1 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier (Accessed 23 June 2023).
- Scikit-learn.org. (n.d.). `sklearn.neighbors.KNeighborsClassifier` — scikit-learn 0.23.1 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier> (Accessed 23 June 2023).
- Scikit-learn.org. (n.d.). 3.2.4.3.1. `sklearn.ensemble.RandomForestClassifier` — scikit-learn 0.21.3 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier> (Accessed 23 June 2023).
- Taboga, M., 2021. Kullback-Leibler divergence Lectures on probability theory and mathematical statistics. Kindle Direct Publishing. Online appendix. Available at: <https://www.statlect.com/fundamentals-of-probability/Kullback-Leibler-divergence>. (Accessed 10 June 2023).

- Tieleman, T., and Hinton, G., 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSEERA: Neural Networks for Machine Learning.
- Taha, A.A. and Malebary, S.J., 2020. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine, *IEEE Access*, IEEE, Vol. 8, pp. 25579–25587.
- Takahashi, S., Chen, Y. and Tanaka-Ishii, K., 2019. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527, 121261.
- Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., and Singh, A. K., 2021. Credit card fraud detection using machine learning: a study. *ArXiv Preprint ArXiv:2108.10005*.
- Vega-Márquez, B., Rubio-Escudero, C., Riquelme, J. C. and Nepomuceno-Chamorro, I., 2020. Creation of Synthetic Data with Conditional Generative Adversarial Networks. In: . 231–240.
- Van Belle, R., Baesens, B. and De Weerd, J., 2023. CATCHM: A novel network-based credit card fraud detection method using node representation learning, *Decision Support Systems*, Elsevier, Vol. 164, p. 113866.
- Vuttipittayamongkol, P. and Elyan, E., 2020. Neighbourhood-based undersampling approach for handling imbalanced and overlapped data, *Information Sciences*, Elsevier, Vol. 509, pp. 47–70.
- Vijayaraghavan, S. and Guan, T., 2022. GAN based Data Augmentation to Resolve Class Imbalance, *ArXiv Preprint ArXiv:2206.05840*.
- Weijis, S. V., Van Nooijen, R., and Van De Giesen, N., 2010. Kullback–Leibler divergence as a forecast skill score with classic reliability–resolution–uncertainty decomposition. *Mon. Weather Rev.* 138, 3387–3399.
- William H Wolberg, W Nick Street, and Olvi L Mangasarian, Breast cancer Wisconsin (diagnostic) data set, UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml/>], 1992.
- William, H., Wolberg, W., Street, N., and Olvi, L. Mangasarian. In UCI Machine Learning Repository; School of Information and Computer Science, University of California: Irvine, CA, USA, 1995; Available online: [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)) (Accessed 06 June 2023).
- Wolpert, D.H., 1992. Stacked generalization, *Neural Networks*, 5(2), pp. 241–259. Available at: [https://doi.org/10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1).
- Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X. and Wang, F.-Y., 2017. Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica* [online], 4 (4), 588–598. Available from: <http://dx.doi.org/10.1109/jas.2017.7510583>.
- Wang, Z., Wang, Y. and Srinivasan, R.S., 2018. A novel ensemble learning approach to support building energy use prediction, *Energy and Buildings*, 159, pp. 109–122.
- Wolterink, J. M., Kamnitsas, K., Ledig, C. and Išgum, I., 2020. Deep learning: Generative adversarial networks and adversarial methods. *Handbook of Medical Image Computing and Computer Assisted Intervention* [online], 547–574. Available from: <http://dx.doi.org/10.1016/b978-0-12-816176-0.00028-4>.
- Wang, K., Zhang, X., Hao, Q., Wang, Y. and Shen, Y., 2019. Application of improved least-square generative adversarial networks for rail crack detection by AE technique. *Neurocomputing* [online], 332, 236–248. Available from: <http://dx.doi.org/10.1016/j.neucom.2018.12.057>.

- Wickramaratne, S. D., and Mahmud, M. S., 2021. Conditional-GAN based data augmentation for deep learning task classifier improvement using fNIRS data. *Frontiers in Big Data*, 4, 659146.
- Wei, J., Huang, H., Yao, L., Hu, Y., Fan, Q. and Huang, D., 2020. NI-MWMOTE: An improving noise-immunity majority weighted minority oversampling technique for imbalanced classification problems, *Expert Systems with Applications*, Elsevier, Vol. 158, p. 113504.
- Werner de Vargas, V., Schneider Aranda, J.A., dos Santos Costa, R., da Silva Pereira, P.R. and Victória Barbosa, J.L., 2023. Imbalanced data preprocessing techniques for machine learning: a systematic mapping study, *Knowledge and Information Systems*, Springer, Vol. 65 No. 1, pp. 31–57.
- Wan, Z., Zhang, Y. and He, H., 2017. Variational autoencoder based synthetic data generation for imbalanced learning. 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 1–7.
- Wang, B.X. and Japkowicz, N., 2004. Imbalanced data set learning with synthetic samples, *Proc. IRIS Machine Learning Workshop*, Vol. 19, p. 435.
- Wang, S., Dai, Y., Shen, J. and Xuan, J., 2021. Research on expansion and classification of imbalanced data based on SMOTE algorithm, *Scientific Reports*, Springer, Vol. 11 No. 1, pp. 1–11.
- Xie, Y., Liu, G., Cao, R., Li, Z., Yan, C. and Jiang, C. 2019. A feature extraction method for credit card fraud detection, 2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS), IEEE, pp. 70–75.
- Xue, W. and Zhang, J., 2016. Dealing with imbalanced dataset: A re-sampling method based on the improved SMOTE algorithm, *Communications in Statistics-Simulation and Computation*, Taylor & Francis, Vol. 45 No. 4, pp. 1160–1172.
- Xue, Y. and Qin, J., 2022. Partial connection based on channel attention for differentiable neural architecture search, *IEEE Transactions on Industrial Informatics*, IEEE.
- Xie, X., Liu, H., Zeng, S., Lin, L. and Li, W., 2021. A novel progressively undersampling method based on the density peaks sequence for imbalanced data, *Knowledge-Based Systems*, Elsevier, Vol. 213, p. 106689.
- Xie, L., Lin, K., Wang, S., Wang, F. and Zhou, J., 2018. Differentially Private Generative Adversarial Network, 1, 1–9.
- Xu, L. and Veeramachaneni, K., 2018. Synthesizing tabular data using generative adversarial networks. *ArXiv Preprint ArXiv:1811.11264*.
- Yang, W., Zhang, Y., Ye, K., Li, L., and Xu, C.-Z. 2019, Ffd: A federated learning based method for credit card fraud detection. *International Conference on Big Data*, 18–32.
- Y. Cao, L. Jia, Y. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X. Li and H. Dai, 2018. Recent Advances of Generative Adversarial Network in Computer Vision, in *IEEE*, pages 1-22, 2018.
- Yu, Z., Lan, K., Liu, Z. and Han, G., 2021. Progressive ensemble kernel-based broad learning system for noisy data classification, *IEEE Transactions on Cybernetics*, IEEE.
- Yang, K., Yu, Z., Chen, C.L.P., Cao, W., Wong, H.-S., You, J. and Han, G., 2021. Progressive hybrid classifier ensemble for imbalanced data, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, IEEE, Vol. 52 No. 4, pp. 2464–2478.
- Zamini, M., and Montazer, G., 2018. Credit card fraud detection using autoencoder based clustering. 2018 9th International Symposium on Telecommunications (IST), 486–491.

- Zheng, W., Yan, L., Gou, C., and Wang, F.-Y., 2021. Federated meta-learning for fraudulent credit card detection. *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4654–4660.
- Zheng, L., Liu, G., Yan, C. and Jiang, C., 2018. Transaction fraud detection based on total order relation and behavior diversity, *IEEE Transactions on Computational Social Systems*, IEEE, Vol. 5 No. 3, pp. 796–806.
- Zhong, J., Li, Y., Xie, W., Lei, J. and Jia, X., 2022. Multi-Prior Twin Least-Square Network for Anomaly Detection of Hyperspectral Imagery. *Remote Sensing* [online], 14 (12), 2859. Available from: <http://dx.doi.org/10.3390/rs14122859>.
- Zhang, Z., Li, M. and Yu, J., 2018. On the convergence and mode collapse of GAN. *SIGGRAPH Asia 2018 Technical Briefs* [online]. Available from: <http://dx.doi.org/10.1145/3283254.3283282>.
- Zioviris, G., Kolomvatsos, K. and Stamoulis, G., 2022. Credit card fraud detection using a deep learning multistage model, *The Journal of Supercomputing*, Springer, pp. 1–26.
- Zhang H., Tang W., Na W., Lee P., Kim J., 2020. Implementation of generative adversarial network-CLS combined with bidirectional long short-term memory for lithium-ion battery state prediction . *Journal of Energy Storage* 31
- Zhang, Z., Yang, L., Chen, L., Liu, Q., Meng, Y., Wang, P. and Li, M., 2020. A generative adversarial network-based method for generating negative financial samples. *International Journal of Distributed Sensor Networks*, 16 (2), 155014772090705.
- Zhang, H. and Li, M., 2014. RWO-Sampling: A random walk over-sampling approach to imbalanced data classification, *Information Fusion*, Elsevier, Vol. 20, pp. 99–116.
- Zhang, R., Zhang, Z. and Wang, D., 2021. RFCL: A new under-sampling method of reducing the degree of imbalance and overlap, *Pattern Analysis and Applications*, Springer, Vol. 24 No. 2, pp. 641–654.
- Zhang, S., 2020. Cost-sensitive KNN classification. *Neurocomputing* 391, 234–242.
- Zhou, Y., Dong, F., Liu, Y., Li, Z., Du, J. and Zhang, L., 2020. Forecasting emerging technologies using data augmentation and deep learning, *Scientometrics*, Springer, Vol. 123, pp. 1–29.

Appendix

Deploying a Synthetic K-CGAN Model for Credit Card

Transactions on AWS Using Flask REST API

In this project, a synthetic K-CGAN model has been trained using credit card transaction data. The model has been developed to generate synthetic credit card transactions that can be used to augment the original dataset for better model training. A Flask REST API has been developed to serve the model and the synthetic transactions generated by it. The API can be accessed through HTTP requests, and the model can be deployed on AWS using services such as Elastic Beanstalk or EC2. The synthetic data can be used to improve the performance of machine learning models while also demonstrating deployment of the model on the cloud using Flask REST API and AWS services.

Below are 3 major sections of this deployment process.

Flask API development

Develop a REST API using the Flask web framework to serve the K-CGAN model and the generated transactions.

Code:

- This is a Flask app that generates synthetic data using a GAN model.
- The app has two endpoints: a default endpoint '/' that returns a message 'Flask is running!', and another endpoint '/generate_synthetic_data' that generates synthetic data and returns it as a JSON response.
- The '/generate_synthetic_data' endpoint expects a POST request with two parameters: 'samples' and 'transaction_type'.
- The 'samples' parameter specifies the number of synthetic data samples to generate.
- The 'transaction_type' parameter specifies the type of transaction to generate, either 'fraud' or 'valid'.
- The GAN model used to generate synthetic data is loaded from a saved model file using the NoveltyGAN class defined in the 'novelty_gan.py' file.
- The 'port' variable is set to the value of the 'PORT' environment variable, or 5000 if the 'PORT' variable is not set.
- The Flask app is started on the host '0.0.0.0' and the specified port, with debug mode set to False.
- Saved model is generator_90, saved in models folder.

Define Novelty K-CGAN class

This class is used to generate the synthetic data.

```
# Load Packages
```

```

from flask import Flask, request, jsonify
import pandas as pd
import tensorflow as tf
import warnings
from sklearn.preprocessing import MinMaxScaler
warnings.filterwarnings('ignore')

class NoveltyGAN:
    """
    A class for generating synthetic tabular data using a Generative Adversarial Network (GAN).

    Parameters:
    -----
    path : str
        The path to the saved generator model.
    dataset : str
        The path to the dataset used for training the GAN.

    Attributes:
    -----
    features : list of str
        The column names of the input dataset used as features.
    standard_scaler : MinMaxScaler
        The scaler object fit to the input dataset for normalizing the data.

    Methods:
    -----
    scaling_data():
        Reads in the dataset, removes the "Time" column, and scales the input data between -1 and 1 using
        MinMaxScaler.
    generator_synthetic_data(samples = 10, transaction_type = 'fraud'):

```

Generates synthetic data using the saved generator model and the specified number of samples and transaction type

```
"""
```

```
def __init__(self, path, dataset):
```

```
    self.dataset = dataset
```

```
    self.path = path
```

```
    self.features = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',  
                    'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',  
                    'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount']
```

```
    self.scaling_data()
```

```
def scaling_data(self):
```

```
    """
```

Reads in the dataset, removes the "Time" column, and scales the input data between -1 and 1 using MinMaxScaler.

```
    """
```

```
    cc_df = pd.read_csv(self.dataset)
```

```
    cc_df = cc_df.drop("Time", axis=1)
```

```
    cc_df = cc_df.astype('float32')
```

```
    standard = MinMaxScaler(feature_range=(-1, 1))
```

```
    self.standard_scaler = standard.fit(cc_df[self.features])
```

```
def generator_synthetic_data(self, samples=10, transaction_type='fraud'):
```

```
    """
```

Generates synthetic data using the saved generator model and the specified number of samples and transaction type.

Parameters:

```
-----
```

`samples` : int, optional

The number of synthetic samples to generate (default=10).

`transaction_type` : str, optional

The type of synthetic transaction to generate, either 'fraud' or 'normal' (default='fraud').

Returns:

synthetic_data : pandas DataFrame

The generated synthetic data, with a "Class" column added to indicate whether the transaction is fraudulent (1) or not (0).

"""

```
generator = tf.keras.models.load_model(self.path)
```

```
noiseG = tf.random.normal(shape=(samples, 100))
```

```
if transaction_type == 'fraud':
```

```
    synthetic_data = generator.predict([noiseG, tf.ones((samples, 1))])
```

```
else:
```

```
    synthetic_data = generator.predict(
```

```
        [noiseG, tf.zeros((samples, 1))])
```

```
synthetic_data = pd.DataFrame(synthetic_data, columns=self.features)
```

```
synthetic_data = pd.DataFrame(self.standard_scaler.inverse_transform(
```

```
    synthetic_data), columns=self.features)
```

```
if transaction_type == 'fraud':
```

```
    synthetic_data["Class"] = 1
```

```
else:
```

```
    synthetic_data["Class"] = 0
```

```
return synthetic_data
```

Flask API

This code is use to build a rest api to communicate with above K-CGAN class and generate synthetic data

```
#Importing necessary modules
```

```
from flask import Flask, request, jsonify
```

```
import pandas as pd
```

```
from novelty_gan import NoveltyGAN # importing NoveltyGAN class
```

```
import os
```

```
#Creating Flask app instance
```

```
app = Flask(name)
```

```

#Loading GAN Model

gan_model = NoveltyGAN('models/generator_90', 'creditcard.csv') # Provide model path and data file
path

#Defining root route of the app

@app.route('/')

def hello():

    return 'Flask is running!' # Return a simple message to check if the app is running or not

# Defining '/generate_synthetic_data' route of the app to generate synthetic data

@app.route('/generate_synthetic_data', methods=['GET'])

def generate_synthetic_data():

    # Get data from the request

    data = request.args

    # Extract the required data from the JSON data

    samples = data['samples']

    transaction_type = data['transaction_type']

    # Use the GAN model to generate synthetic data

    synthetic_data = gan_model.generator_synthetic_data(samples, transaction_type)

    # Convert the generated data to JSON format

    response = synthetic_data.to_json(orient='records')

    return response # Return the JSON formatted data as response to the request

#Start the Flask app if this file is run directly

if name == 'main':

    # Get the port from the environment variable, or set it to 5000 by default

    port = int(os.environ.get('PORT', 5000))

    # Run the app on 0.0.0.0 IP address (to make it accessible from outside the container) and the specified
port

    app.run(host='0.0.0.0', port=port, debug=False) # Turn debug off in production

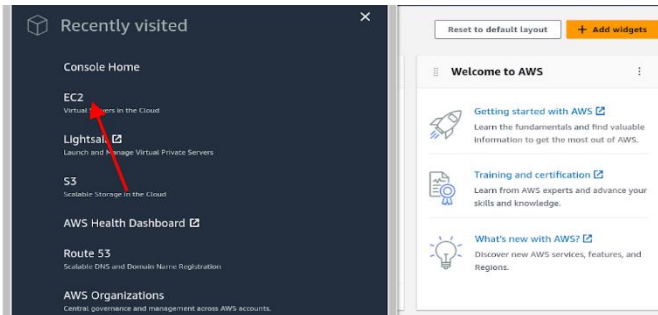
```

AWS development

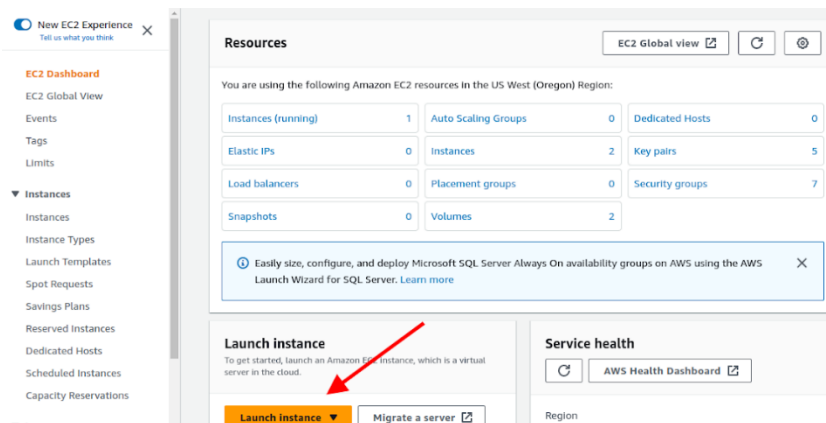
Deploy the Flask API and the GAN model on the AWS cloud using EC2. The steps are divided in two sub sections.

Signup on AWS and Get Instance

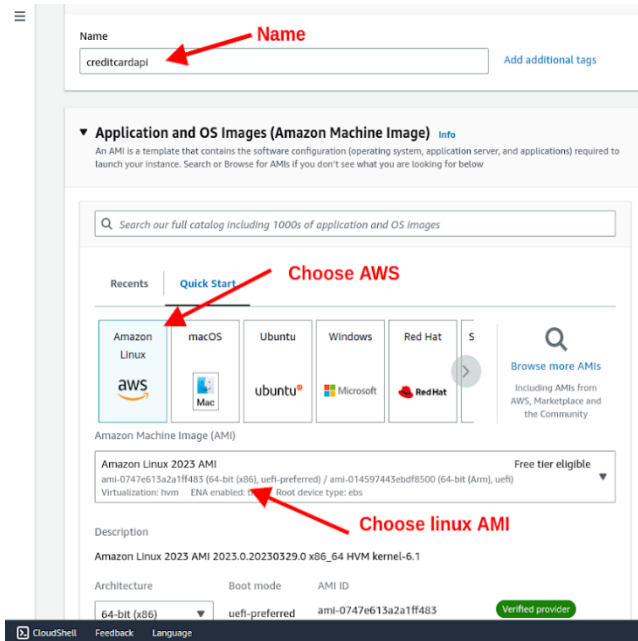
1. Navigate to the [AWS website](#) and click on the **"Create an AWS Account"** button.
2. Follow the steps to create your account. You will need to provide your email address, create a password, and enter your billing information.
3. Once the account is created, sign in to the [AWS Management Console](#).
4. Click on the "EC2" service to launch the EC2 dashboard.



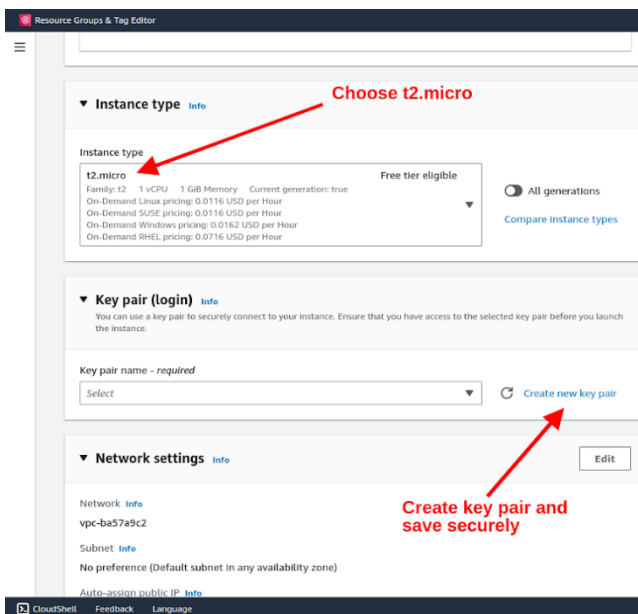
5. Click on the "Launch Instance" button to start the process of creating a new EC2 instance.



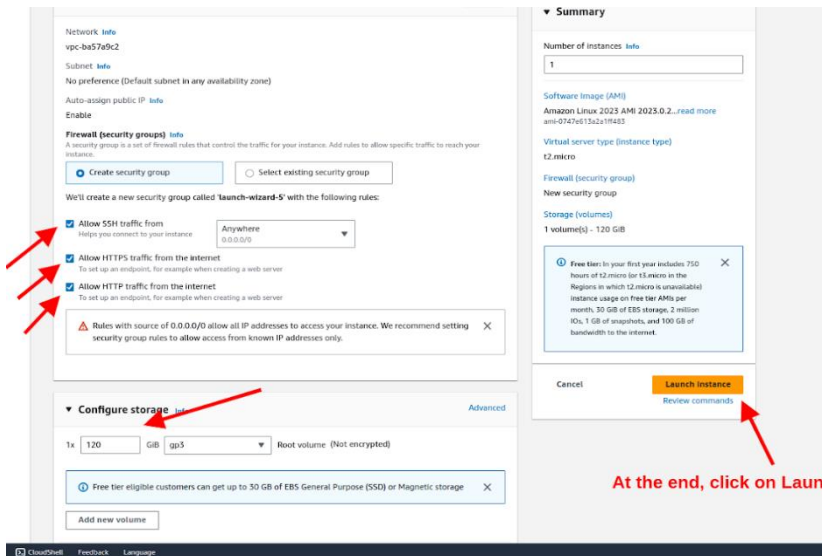
6. Select the "Amazon Linux 2 AMI (HVM), SSD Volume Type" as the AMI (Amazon Machine Image) for the instance.



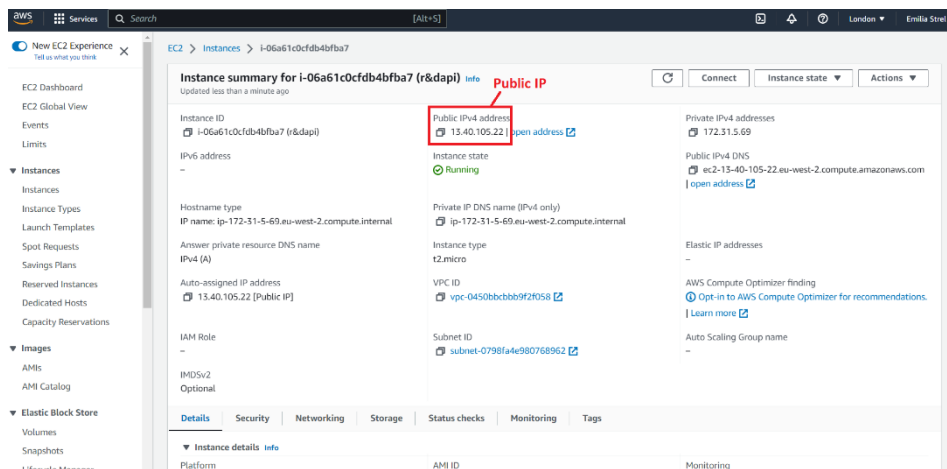
7. Select the "t2.micro" instance type from the list of available instance types.



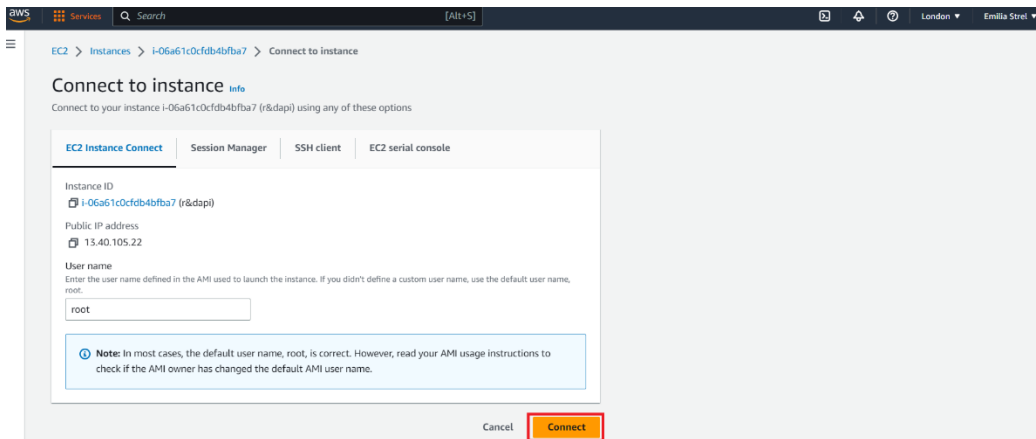
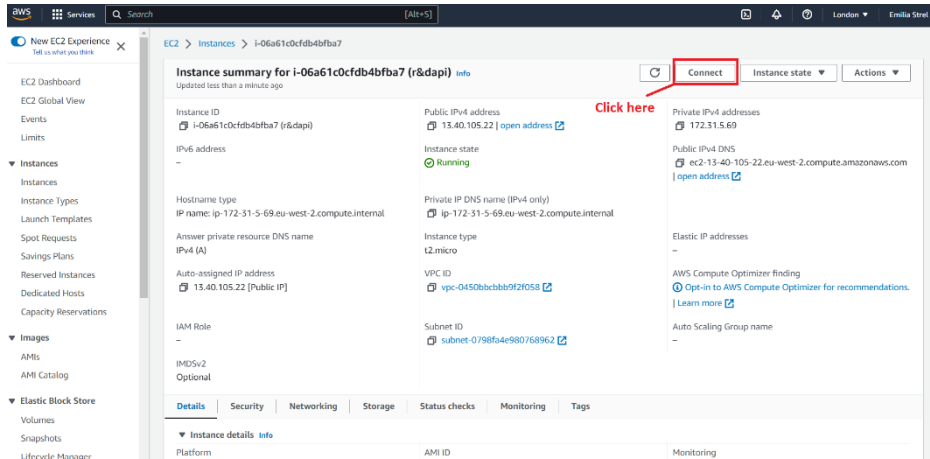
8. Configure instance details, including the number of instances, network settings, and storage.
9. Select or create a security group for the instance. A security group controls the traffic to and from your instance.
10. Review instance settings and launch the instance.



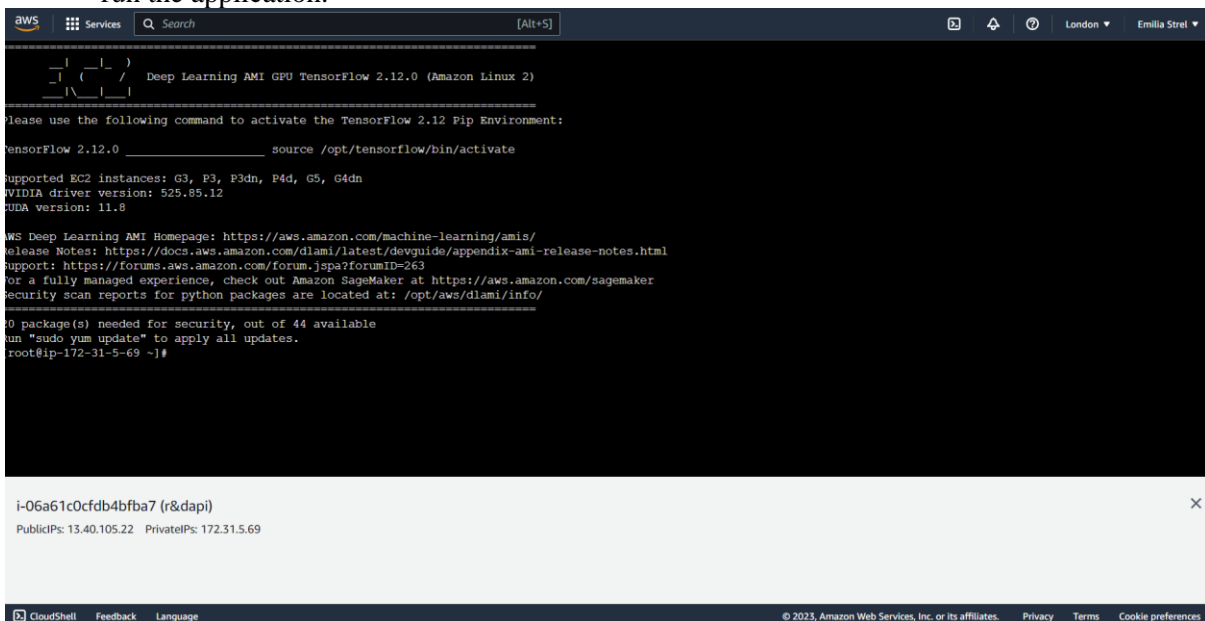
11. Download the private key file (.pem) for the instance.
12. Connect to the instance using SSH by running the following command in the terminal:
`ssh -i /path/to/private/key.pem ec2-user@<your-instance-public-ip>`



13. Then click on Connect in the next screen.



14. Once connected to the instance, install and configure any necessary software, including Flask, to run the application.



Bellow commands could be utilized to upload and run FLASK APP.

Steps to Upload FLASK APP

1. Access terminal or command prompt and navigate to the directory where the Flask app is located.
2. Create a new SSH key pair by running the following command in the terminal:
`ssh-keygen -t rsa -b 4096`

This will create a new SSH key pair in the `~/.ssh` directory on the local machine.

3. Log in to the AWS Management Console, navigate to the EC2 dashboard, and launch a new EC2 instance. Select the `t2.micro` instance type and choose the appropriate security group and key pair settings.
4. Once the EC2 instance is up and running, SSH into it by running the following command in the terminal:
`ssh -i /path/to/your/key.pem ec2-user@34.210.202.193`

Replace `/path/to/your/key.pem` with the path to your SSH key file, and `ec2-user` with the appropriate username for the EC2 instance.

5. Install the necessary packages and dependencies on the EC2 instance by running the following commands:
`sudo yum update`
`sudo yum install python3`
`sudo yum install python3-pip`
`sudo pip3 install flask pandas scikit-learn tensorflow`

6. Use filezilla to transfer the files or copy Flask app files to the EC2 instance by running the following command:

```
scp -i /path/to/your/key.pem /path/to/your/flask/app ec2-user@13.40.105.22:/home/ec2-user
```

Replace `/path/to/your/flask/app` with the path to your Flask app directory.

7. SSH into the EC2 instance again and navigate to the Flask app directory:
`ssh -i /path/to/your/key.pem ec2-user@13.40.105.22`
`cd /home/ec2-user/path/to/your/flask/app`
8. Start the Flask app by running the following command:
`nohup python main.py &`

9. This will start the Flask app on the public IP address of the EC2 instance.

10. Test the Flask app by navigating to <http://13.40.105.22:5000> in the web browser.

Upon performing the above steps the Flask app should now be up and running on the AWS EC2 instance.

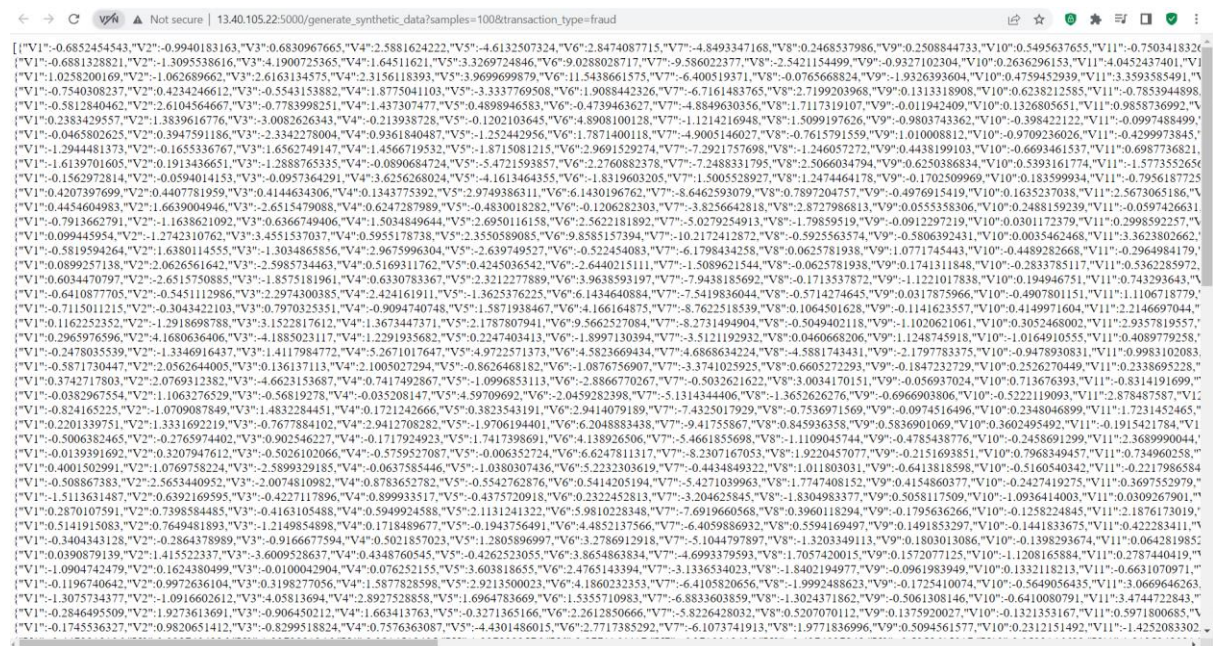
API Testing

Test the API to ensure that it is functioning correctly and that the synthetic transactions generated by the K-CGAN model are of high quality.

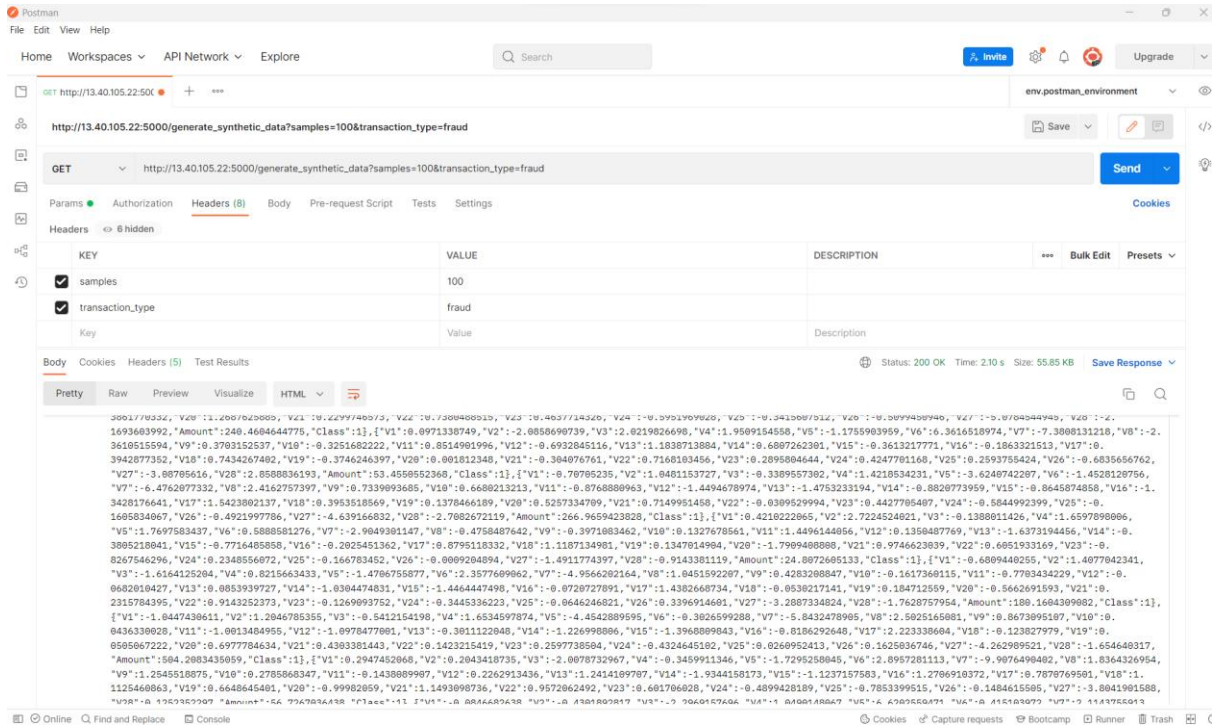
The steps below demonstrate how to test the REST API using Postman:

1. Install [Postman](#) on the system
2. Launch Postman and create a new request by clicking on the "New" button.
3. Select the HTTP method as "GET" from the dropdown list.
4. Enter the URL of the Flask app with the endpoint `/generate_synthetic_data`. For example, if you are running the app locally on port 5000, the URL would be `http://13.40.105.22:5000/generate_synthetic_data`.
5. In the "Params" tab, enter the following two key-value pairs in the "KEY" and "VALUE" columns respectively: "samples" and the number of samples you want to generate, and "transaction_type" and the type of transaction for which you want to generate synthetic data.
6. Click on the "Send" button to send the request to the Flask app.
7. You will receive a response in the JSON format with the generated synthetic data.

Link - http://13.40.105.22:5000/generate_synthetic_data?samples=100&transaction_type=fraud

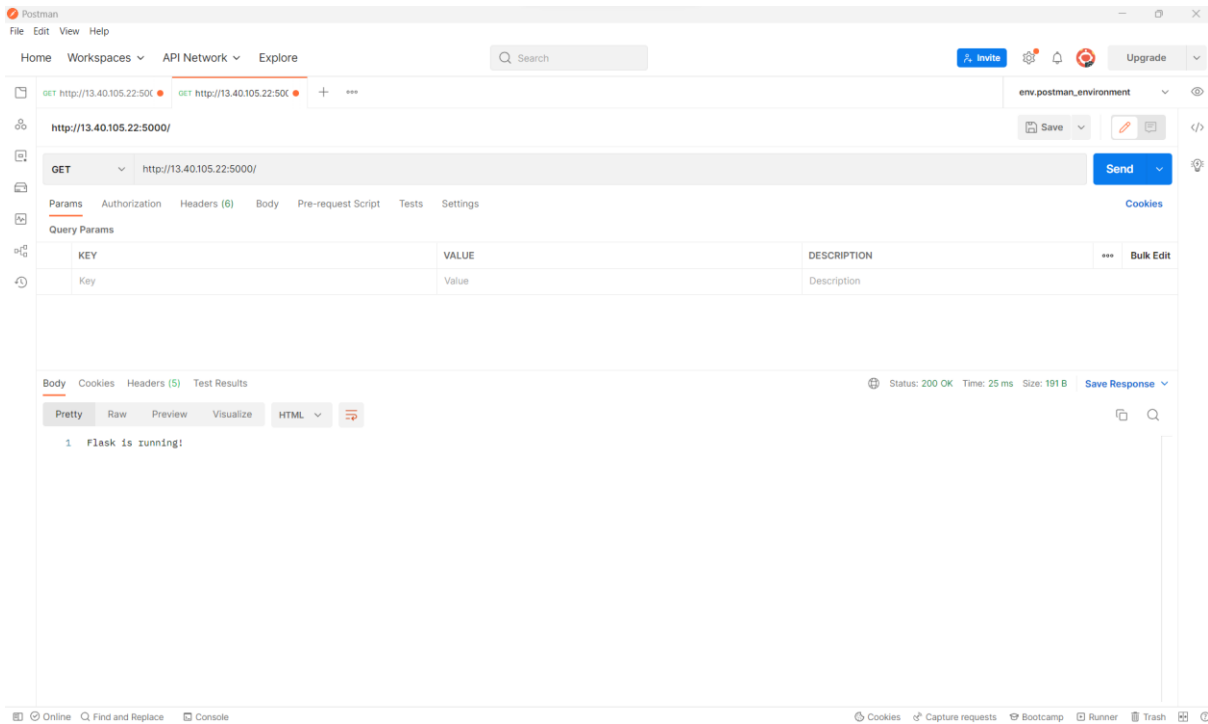


```
[{"V1": -0.6852454543, "V2": -0.9940183163, "V3": 0.6830967665, "V4": 2.5881624222, "V5": -4.6132507324, "V6": 2.8474087715, "V7": -4.8493347168, "V8": -0.2468537986, "V9": -0.2508844733, "V10": -0.5495637655, "V11": -0.7503418326, "V12": -0.6881328821, "V13": -1.3095538616, "V14": -4.1900725365, "V15": 1.64511621, "V16": -3.3269724846, "V17": 9.0288028717, "V18": -9.586022377, "V19": -2.5421154499, "V20": -0.9327102304, "V21": 0.263296153, "V22": -0.0452437401, "V23": 1.0258200169, "V24": -1.062689662, "V25": 2.6163134575, "V26": 2.3156118393, "V27": 3.9699699879, "V28": 11.5438661575, "V29": -6.400519371, "V30": -0.0765668824, "V31": -1.9326393604, "V32": -0.4759452939, "V33": 3.593585491, "V34": -0.7540308237, "V35": -0.4234246612, "V36": -0.5545135882, "V37": 1.8775041103, "V38": -3.3337769508, "V39": -1.9088442326, "V40": -6.7161483765, "V41": 2.7199203968, "V42": -0.1313318908, "V43": 0.6238212585, "V44": -0.7853944898, "V45": -0.5812840462, "V46": 2.6104564667, "V47": -0.7783998251, "V48": 1.4373074777, "V49": -0.4898946583, "V50": -0.4739463627, "V51": -4.8849630356, "V52": 1.7117319107, "V53": -0.011942409, "V54": -0.9803743362, "V55": -0.1326805651, "V56": -0.9858736992, "V57": 0.2383429557, "V58": 1.3839616776, "V59": -3.0082626343, "V60": -0.213938728, "V61": -0.1202103645, "V62": 4.8908100128, "V63": -1.1214216948, "V64": -1.5099197626, "V65": -0.9803743362, "V66": -0.398422122, "V67": -0.0997484849, "V68": -0.0465802625, "V69": -0.3947591186, "V70": -2.3342278004, "V71": 0.9361840487, "V72": -1.252442956, "V73": 1.7871400118, "V74": -4.9005146027, "V75": -0.7615791559, "V76": 1.01008812, "V77": -0.9709236026, "V78": -0.4299973845, "V79": -1.294481373, "V80": -0.1655336767, "V81": 1.6562749147, "V82": 1.4566719532, "V83": -1.8715801215, "V84": 2.9691529274, "V85": -7.2921757698, "V86": -1.246057272, "V87": -0.4438199103, "V88": -0.6693461537, "V89": -0.6987736821, "V90": -1.6139701605, "V91": 0.1913436651, "V92": -1.2888765335, "V93": -0.0890684724, "V94": -0.5955178738, "V95": -5.4721593857, "V96": 2.2760882378, "V97": -7.2488331795, "V98": -2.5066034794, "V99": -0.6250386834, "V100": -0.5393161774, "V101": -1.5773552656, "V102": -0.1562972814, "V103": -0.0594914153, "V104": -0.0957364291, "V105": 3.6256268024, "V106": -4.1613464355, "V107": 1.8319603205, "V108": 1.5005528927, "V109": 1.2474464178, "V110": -0.1702509969, "V111": 0.1835999934, "V112": -0.7956187725, "V113": -0.4207397699, "V114": 0.4407781959, "V115": 0.4146434306, "V116": 0.1343775392, "V117": -0.9749386311, "V118": 6.1430196762, "V119": -8.646259079, "V120": 0.7897204757, "V121": 0.6163237038, "V122": 2.5673065186, "V123": 0.4454604983, "V124": 1.6639004946, "V125": 2.6315479088, "V126": 0.6247287989, "V127": -0.4830018382, "V128": -0.1206232303, "V129": -3.8256642818, "V130": 2.8727986813, "V131": -0.055538306, "V132": -0.2488519239, "V133": -0.0597426631, "V134": -0.7913662791, "V135": -1.1638621092, "V136": -0.6366749406, "V137": 1.5034849644, "V138": 2.6950116158, "V139": 2.5622181892, "V140": -5.0279254913, "V141": -1.79859519, "V142": -0.0912297219, "V143": -0.0301172379, "V144": 0.2998592257, "V145": 0.099445954, "V146": -1.2742310762, "V147": 3.4551537037, "V148": 0.5955178738, "V149": 2.3505898085, "V150": 9.8585157394, "V151": -10.2172412872, "V152": -0.5925563574, "V153": -0.5806392431, "V154": -0.0035462468, "V155": 3.3623802662, "V156": 0.5819594264, "V157": 1.6380114555, "V158": -1.3034865856, "V159": 2.9675996304, "V160": -2.639749527, "V161": -0.522454083, "V162": 6.1798434258, "V163": -0.0625781938, "V164": 0.0771745443, "V165": -0.4489282668, "V166": -0.2964984179, "V167": 0.0899257138, "V168": 2.0626561642, "V169": -2.5985734463, "V170": 0.5169311762, "V171": -0.4245036542, "V172": 2.6440215111, "V173": -1.5089621544, "V174": -0.0625781938, "V175": 0.174131848, "V176": -0.2833785117, "V177": -0.5362285972, "V178": 0.6034470797, "V179": -2.6515758085, "V180": -1.8575181961, "V181": 0.6330783367, "V182": 2.3212277889, "V183": 3.9638593197, "V184": -7.9438185692, "V185": -0.1713537872, "V186": -1.1221017838, "V187": -0.1949467551, "V188": -0.743293643, "V189": -0.641087705, "V190": -0.5451112986, "V191": 2.2297400385, "V192": 2.424161911, "V193": -1.3625376225, "V194": 6.1434640884, "V195": -7.5419836044, "V196": -0.5714274645, "V197": -0.0317875966, "V198": -0.4907801151, "V199": -1.1162718779, "V200": -0.7115011215, "V201": -0.3043422103, "V202": 0.7970325531, "V203": -0.9094740748, "V204": -1.5871938467, "V205": 4.166164875, "V206": -8.7622518539, "V207": -0.1064501628, "V208": -0.1141623557, "V209": -0.4149971604, "V210": -2.2146697044, "V211": 0.1162252352, "V212": -1.2918698788, "V213": 3.1522817612, "V214": 1.3673447371, "V215": 2.1787807941, "V216": 9.5662527084, "V217": -8.2731494904, "V218": -0.5049402118, "V219": -1.1020621061, "V220": 0.3052468002, "V221": 2.9357819557, "V222": -0.2965976596, "V223": -4.1680636406, "V224": -4.1885023117, "V225": 1.2291935682, "V226": -0.2247404413, "V227": 1.8997130394, "V228": -3.5121192932, "V229": -0.0460668206, "V230": 1.1248745918, "V231": -1.0164910555, "V232": -0.4089779258, "V233": -0.2478035339, "V234": -1.3346916437, "V235": 1.4117984772, "V236": 5.2671017647, "V237": 4.9722571373, "V238": -4.5323669434, "V239": 4.6868634224, "V240": -4.5881743431, "V241": -2.1797783375, "V242": -0.9478930831, "V243": -0.9983102083, "V244": -0.5871730447, "V245": 2.0562644005, "V246": 0.136137113, "V247": 2.1003027294, "V248": -0.8626468182, "V249": -1.0876756907, "V250": -3.3741025925, "V251": -0.6605272293, "V252": -0.1847232729, "V253": -0.2536270449, "V254": -0.238695228, "V255": 0.3742717803, "V256": 0.0769312382, "V257": -4.6623153687, "V258": -0.7417492867, "V259": -1.0998653113, "V260": -2.8866770267, "V261": -0.5032621622, "V262": 3.0034170151, "V263": -0.056937024, "V264": 0.71367693, "V265": -0.8314191699, "V266": -0.0382967554, "V267": 1.0632765828, "V268": -0.56819278, "V269": -0.035208147, "V270": 4.59709692, "V271": -2.0459282398, "V272": 5.1314344406, "V273": -1.3652626276, "V274": -0.6966903806, "V275": -0.522210609, "V276": 2.878487587, "V277": -0.824165225, "V278": -1.0790987849, "V279": 1.4832284451, "V280": 0.1721242666, "V281": -0.3823543191, "V282": 2.9414079189, "V283": -7.4325017929, "V284": -1.3652626276, "V285": -0.0974516496, "V286": -0.2348046899, "V287": 1.7231452465, "V288": 0.2201339751, "V289": 1.3331692219, "V290": -0.7677884102, "V291": 2.9412708282, "V292": -1.9706194401, "V293": 6.2048883438, "V294": -9.41755867, "V295": -0.845936358, "V296": 0.5836091069, "V297": -0.3602495492, "V298": -0.1915421784, "V299": -0.5006382465, "V300": -0.2765974402, "V301": -0.902546227, "V302": -0.1717924923, "V303": -1.7417938691, "V304": 6.138926506, "V305": -5.4661855698, "V306": -1.1109045744, "V307": -0.4785438776, "V308": -0.2458891299, "V309": 2.3689990044, "V310": -0.013991692, "V311": 0.3207947612, "V312": -0.5026102066, "V313": -0.5759527087, "V314": -0.006352724, "V315": 6.6247811317, "V316": -8.2307167053, "V317": 1.9220457077, "V318": -0.2151693851, "V319": -0.0768349457, "V320": -0.734960258, "V321": 0.4001502991, "V322": 1.0769758224, "V323": -2.589939185, "V324": -0.0637585446, "V325": -1.0380307436, "V326": 5.2232030619, "V327": -0.4434849322, "V328": 1.011803031, "V329": -0.6413818598, "V330": -0.5160540342, "V331": -0.2217986584, "V332": -0.508867383, "V333": 2.5634409257, "V334": -0.0074810982, "V335": -0.5542762878, "V336": 0.544205194, "V337": -5.4271039963, "V338": 1.7747408152, "V339": -0.5427419275, "V340": 0.369752979, "V341": -1.5113631487, "V342": -0.6392169595, "V343": -0.4227117896, "V344": 0.899933517, "V345": -0.4375720916, "V346": 0.2322425813, "V347": -3.204625845, "V348": 1.8304983377, "V349": 0.5058117509, "V350": -1.0936444003, "V351": -0.0309267901, "V352": 0.2870107591, "V353": 0.7395848483, "V354": -0.4163105488, "V355": 0.5949924588, "V356": 2.1131241322, "V357": 9.981023848, "V358": 7.6919660568, "V359": 0.3960118294, "V360": -1.795636266, "V361": -0.125824845, "V362": 2.1876173019, "V363": 0.5141943018, "V364": 0.7649481893, "V365": -1.2149854898, "V366": -0.1718489677, "V367": -0.1943756491, "V368": 4.4852137566, "V369": -6.4059886932, "V370": -0.5594169497, "V371": -0.1491853297, "V372": -0.1441833675, "V373": -0.422283411, "V374": -0.3404343128, "V375": -0.2864378989, "V376": -0.9166677594, "V377": 0.5021857023, "V378": -1.2805896997, "V379": 3.2786912918, "V380": -5.1044797897, "V381": -1.3203349113, "V382": 0.1803013086, "V383": -0.1398293674, "V384": 0.0642819852, "V385": 0.0390879139, "V386": 1.415522337, "V387": 3.6009528637, "V388": 0.4348760545, "V389": -0.4262523055, "V390": 3.8654863834, "V391": -4.6993379593, "V392": 1.7057420015, "V393": 0.1572077125, "V394": -1.1298165884, "V395": 0.2787440419, "V396": -1.0904724279, "V397": 0.1624380499, "V398": -0.0100042904, "V399": 0.076252155, "V400": 3.6038186055, "V401": 2.4765143394, "V402": -3.1336534023, "V403": -1.8402194977, "V404": -0.0961983949, "V405": 0.1332118213, "V406": 0.0661070971, "V407": -1.1196740642, "V408": 0.9972636104, "V409": 0.3198277056, "V410": 1.5877828598, "V411": 2.9213500023, "V412": 1.5355710983, "V413": 6.4105820656, "V414": -1.9992488623, "V415": -0.1725410074, "V416": -0.5649056435, "V417": -0.6630646263, "V418": -1.3075734377, "V419": -0.1091602612, "V420": -0.05813694, "V421": 2.8927528858, "V422": -0.6964783669, "V423": 1.5355710983, "V424": -6.8833603859, "V425": -3.024371862, "V426": 0.5061308146, "V427": -0.6410800791, "V428": 3.4744722843, "V429": -0.2846495509, "V430": 1.9273613691, "V431": -0.906450212, "V432": 1.663413763, "V433": -0.3271365166, "V434": 2.2612850666, "V435": -5.8226428032, "V436": -0.5207070112, "V437": 0.1375920027, "V438": -0.1321353167, "V439": 0.5971800685, "V440": -0.1745536327, "V441": -0.9820651412, "V442": -0.8299518824, "V443": 0.7576365087, "V444": -4.4301486015, "V445": 2.717385292, "V446": -6.1073741913, "V447": 1.9771836996, "V448": -0.5094561577, "V449": -0.2312151492, "V450": -1.4252083302}
```

8. Its also possible to test the root endpoint by selecting the HTTP method as "GET" and entering the URL of the Flask app without any endpoint. For example, if the up is running locally on port 5000, the URL would be "<http://13.40.105.22:5000/>".

URL - <http://13.40.105.22:5000/>



Awards

IAER
Accelerating Education through Excellence in Research
INTERNATIONAL ASSOCIATION FOR EDUCATORS AND RESEARCHERS
Accelerating Education through Excellence in Research

IEEE International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications 2022 (IEEE CoNTESA '22)
Mother Teresa University, North Macedonia
15th – 16th December, 2022

Best Paper Award

Comparative Analysis of Machine Learning Algorithms using GANs through Credit Card Fraud Detection

Emilija Strelcenia and Simant Prakoonwit

Simant
Prof. Mahdi H. Miraz, CEng
Publication Chair

Maaruf Ali
Prof. Maaruf Ali, CEng
Conference Chair



INTERNATIONAL ASSOCIATION FOR EDUCATORS AND RESEARCHERS
Accelerating Education through Excellence in Research

IEEE International Conference on Computing, Networking,
Telecommunications & Engineering Sciences Applications 2022
(IEEE CoNTESA '22)

Mother Teresa University, North Macedonia
15th - 16th December, 2022

Certificate of Paper Presentation Awarded to
Emilija Strelcenia

Comparative Analysis of Machine Learning Algorithms using GANs through
Credit Card Fraud Detection

Prof. Maaruf Ali, CEng
Conference Chair

Prof. Mahdi H. Miraz, CEng
Publication Chair

IEEE 2022 The 3rd International Conference on
Computers and Artificial Intelligence Technologies

Quzhou, Zhejiang, China,
November 04-06

Certificate

FOR EXCELLENT ORAL PRESENTATION

The Chairman & the Board of Directors of
CAIT 2022 Conference Certify that

Emilija Strelcenia

Bournemouth University, England

Participated in 2022 the 3rd International Conference on
Computers and Artificial Intelligence Technologies
Your oral presentation has been selected as the best one of this
conference



*IEEE 2022 The 3rd International Conference on Computers and
Artificial Intelligence Technologies*

CERTIFICATE OF APPRECIATION

THIS CERTIFICATE IS AWARDED TO


Emilija Strelscenia

Bournemouth University, England

Generating synthetic data for credit card fraud detection using GANs (C2027)

For your excellent Oral Presentation at the conference and your significant contribution to the success of the 3rd International Conference on Computers and Artificial Intelligence Technologies (CAIT 2022), be held in Quzhou, Zhejiang, China, during November 04-06, 2022.

Prof. Zhuoran Wang
Prof. Zhuoran Wang
Conf. Chair
CAIT NET

A circular blue logo for the 3rd International Conference on Computers and Artificial Intelligence Technologies (CAIT 2022). The logo features a star at the top and the acronym 'CAIT' in the center. The full name of the conference is written around the perimeter of the circle.



CERTIFICATE



This is proudly certificated that

Emilija Strelcenia

Bournemouth University, UK

Title :

Application of a novel generative adversarial network K-CGAN to industrial maintenance: A case study of Electrical Fault detection and classification(ES2008-A)

whose presentation has been selected as excellent oral presentation on 2022 the 12th International Conference on Power and Energy Systems (ICPES 2022) held in Guangzhou, China during December 23-25, 2022

Co-sponsored by



Conference Committee

CERTIFICATE

This is proudly certified that

Emilija Strelcenia

Bournemouth University, UK

Paper ID: ES2008-A

Paper Title: Application of a novel generative adversarial network K-CGAN to industrial maintenance: A case study of Electrical Fault detection and classification

who has delivered oral presentation on 2022 the 12th International Conference on Power and Energy Systems (ICPES 2022) held in Guangzhou, China during December 23-25, 2022

Co-sponsored by



Conference Committee

Certificate of Participation

This is to certify that

Emilija Strelcenia

hereby recognized for participation and giving an oral presentation in

**2022 International Conference on Digital Management, Information Science and
Technologies (DMIST2022) on March 17, 2023**

Sponsored by:



DMIST2022
Organization Committee

Certificate of Participation

This is to certify that

Emilija Streccenia
presented their research during
The 14th Annual Postgraduate Research
Conference Wednesday 30 November 2022

Signed



Dr Fiona Knight
Head of the Doctoral College



Dr Julia Taylor
Head of the Doctoral College