

Faculty of Science and Technology
Bournemouth University

A thesis submitted in partial fulfilment for the degree of doctor of philosophy



Efficient Multi-Objective NeuroEvolution in Computer Vision and Applications for Threat Identification

Doctoral Thesis of
Daniel Dimanov

Supervisory Team:
Dr. Emili Balaguer-Ballester
Dr. Colin Singleton
Dr. Shahin Rostami

Abstract

Concealed threat detection is at the heart of critical security systems designed to ensure public safety. Currently, methods for threat identification and detection are primarily manual, but there is a recent vision to automate the process. Problematically, developing computer vision models capable of operating in a wide range of settings, such as the ones arising in threat detection, is a challenging task involving multiple (and often conflicting) objectives.

Automated machine learning (AutoML) is a flourishing field which endeavours to discover and optimise models and hyperparameters autonomously, providing an alternative to classic, effort-intensive hyperparameter search. However, existing approaches typically show significant downsides, like their (1) high computational cost/greediness in resources, (2) limited (or absent) scalability to custom datasets, (3) inability to provide competitive alternatives to expert-designed and heuristic approaches and (4) common consideration of a single objective. Moreover, most existing studies focus on standard classification tasks and thus cannot address a plethora of problems in threat detection and, more broadly, in a wide variety of compelling computer vision scenarios.

This thesis leverages state-of-the-art convolutional autoencoders and semantic segmentation (Chapter 2) to develop effective multi-objective AutoML strategies for neural architecture search. These strategies are designed for threat detection and provide insights into some quintessential computer vision problems. To this end, the thesis first introduces two new models, a practical Multi-Objective Neuroevolutionary approach for Convolutional Autoencoders (MONCAE, Chapter 3) and a Resource-Aware model for Multi-Objective Semantic Segmentation (RAMOSS, Chapter 4). Interestingly, these approaches reached state-of-the-art results using a fraction of computational resources required by competing systems (0.33 GPU days compared to 3150), yet allowing for multiple objectives (e.g., performance and number of parameters) to be simultaneously optimised. This drastic speed-up was possible through the coalescence of neuroevolution algorithms with a new heuristic technique termed Progressive Stratified Sampling. The presented methods are evaluated on a range of benchmark datasets and then applied to several threat detection problems, outperforming previous attempts in balancing multiple objectives.

The final chapter of the thesis focuses on thread detection, exploiting these two mod-

els and novel components. It presents first a new modification of specialised proxy scores to be embedded in RAMOSS, enabling us to further accelerate the AutoML process even more drastically while maintaining avant-garde performance (above 85% precision for SIXray). This approach rendered a new automatic evolutionary Multi-objective method for cOncealed Weapon detection (MEOW), which outperforms state-of-the-art models for threat detection in key datasets: a gold standard benchmark (SixRay) and a security-critical, proprietary dataset.

Finally, the thesis shifts the focus from neural architecture search to identifying the most representative data samples. Specifically, the Multi-objective Core-set Discovery through evolutionAry algorithmMs in computEr vision approach (MIRA-ME) showcases how the new neural architecture search techniques developed in previous chapters can be adapted to operate on *data space*. MIRA-ME offers supervised and unsupervised ways to select maximally informative, compact sets of images via dataset compression. This operation can offset the computational cost further (above 90% compression), with a minimal sacrifice in performance (less than 5% for MNIST and less than 13% for SIXray).

Overall, this thesis proposes novel *model-* and *data-centred* approaches towards a more widespread use of AutoML as an optimal tool for architecture and coresets discovery. With the presented and future developments, the work suggests that AutoML can effectively operate in real-time and performance-critical settings such as in threat detection, even fostering interpretability by uncovering more parsimonious optimal models. More widely, these approaches have the potential to provide effective solutions to challenging computer vision problems that nowadays are typically considered unfeasible for AutoML settings.

Dissertation Declaration

This Dissertation/Project Report is submitted in partial fulfilment of the requirements for a PhD degree at Bournemouth University. I declare that this Dissertation/ Project Report is my own work and that it does not contravene any academic offence as specified in the University's regulations.

Retention

I agree that should the University wish to retain it for reference purposes, a copy of my thesis may be held by Bournemouth University normally for a period of 3 academic years. I understand that my thesis may be destroyed once the retention period has expired. I am also aware that the University does not guarantee to retain this thesis for any length of time (if at all) and that I have been advised to retain a copy for my future reference.

Confidentiality

I confirm that this thesis does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular, any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or personal life has been anonymised unless permission for its publication has been granted from the person to whom it relates.

Copyright

The copyright for this thesis remains with me.

Requests for Information

I agree that this thesis may be made available as the result of the Freedom of Information Act.

Signed:



Name: Daniel Dimanov

Date: October 18, 2023

Programme: Doctor of Philosophy(PhD) in Efficient Multi-Objective NeuroEvolution in Computer Vision and Applications for Threat Identification

Original Work Declaration

This thesis and the accompanying artefacts (code) are my own work, except where stated, in accordance with University regulations.

This thesis is a product of my own work, but it would not have been possible without the support and guidance of my supervisory team. Throughout the thesis, the use of "we" and "our" acknowledges the contributions of the other members of the team. It is a way to express my gratitude for their countless hours of proofreading, suggesting improvements, providing computational resources and machines, and offering general direction, mentorship and counselling. The supervisory team is also co-authoring all papers produced as part of this thesis, and I believe it is only fair to recognise their contribution in this way.

Signed: 

Acknowledgements

I would like to begin by thanking my supervisory team, Dr. Emili Balaguer-Ballester, Dr. Shahin Rostami, and Dr. Colin Singleton, for their unwavering support and guidance throughout my PhD journey. Their valuable feedback, timely responses, and willingness to help with whatever they could have been instrumental in my success.

Emili's trust and passion for the project have also been key for submitting papers, proofreading (even on weekends and holidays) and providing general counsel constantly throughout the PhD. Emili is someone I'm honoured to call a friend and has been an emotional and academic bedrock that has always approached the limitless challenges presented to us with resolve and certainty that we will persevere and everything will be alright. He has been the one to stop me from naming my chapters and sections in an even more ridiculous way (yes, I mean it), and I have learnt so much from working closely with such a warm, funny, welcoming, yet hard-working and punctual person.

Colin, who has generously sponsored this PhD project through his company CountingLab has provided a machine that has acted as my best tool during the whole journey as well as a ton of practical applications the PhD work can be applied for. However, this is only a tiny portion of all the deeds and support he has selflessly offered throughout the journey. Having shared over 120 supervisory meetings discussing PhD-related work as well as industry problems has been of extreme importance to keep me on track and deliver some incremental improvement week after week. His feedback and advice have served as invaluable life lessons.

Shahin was tasked with the challenge of picking the right candidate for this PhD project, and I can only thank him and hope that entrusting it to me has proven to be the right choice. Even though he had to depart from the university, he has remained involved with the project and has helped me overcome countless emotional breakdowns and imposter syndrome periods. He has been there when I have needed him most and his entrepreneurial and hard-working spirit has inspired me.

My brother, Dr Bobby, has also been a constant source of support and comfort throughout my PhD. His advice and help have been invaluable, and I am grateful for his willingness to be there for me whenever I needed him. I have jokingly said that he is an unofficial part of the supervisory team, but as much as this is a joke, it is also the truth. He has given me invaluable advice, direction, guidance, loving support, and mul-

tiple once-in-a-lifetime opportunities to join his astonishing ventures before they flourish into undeniable success.

I would also like to thank my parents, Daniela and Todor, for their unconditional love and support. Raising me has never been easy, and given my early disinterest in academia, this whole PhD journey feels like a fever dream for me, but undoubtedly even more for them. Their emotional and financial backing has been crucial in my ability to complete my PhD, and I am grateful for everything they have done for me. Their academic success has been truly inspirational, and both of them have supported me through countless selfless acts. No words can describe how I feel, and while I can't ever repay them for all they have done, I can only hope to bring a smile or tear of joy to their face as, hopefully, their second son also becomes a "fake doctor".

I would also like to thank the Doctoral College at BU and all staff for their support and assistance throughout the years. Naomi Bailey in particular has always kindly responded calmly and with a solution even when I have messed something up spectacularly. The responsiveness, support, and genuine interest in the research of not only me but all PGRs that she is tasked to manage is astonishing, and I'm incredibly grateful for everything she and her colleagues have done.

I would like to thank Dr Gernot Liebchen, who has provided invaluable input during multiple existential crises and has also acted as a supervisory figure as well as a friend every time I have needed his help.

I want to thank my peers and fellow PGRs, especially Rushan Arshad and Kevin Wilson, for sharing this journey with me, giving me unforgettable memories, and making the journey a lot more enjoyable.

I would also like to extend my gratitude to my examiners, **Dr. Marcin Budka** and **Dr. José D. Martin-Guerrero**, for agreeing to take the time to read, review, examine, and provide constructive criticism of my work. Their feedback has been invaluable, and I am grateful for their expertise and guidance. Thank you for your contribution to the successful completion of my PhD.

Last but definitely not the least, I would like to thank my girlfriend, Kosara, for her love and support throughout the whole PhD journey. Her love and care have been a source of comfort, strength and motivation during the good times, the bad times, the mental breakdowns, the sickness and everything in between during the PhD. She is the one that has pushed me countless times and encouraged me with the much-needed "You can do it!". More importantly, even when I have felt that everything is against me or nothing is working and I have almost stopped believing, she would come and look with me at the infinitely fast spinning numbers of a neural network training with a lower batch size and cheer with me the algorithms as they beat the state-of-the-art in front of our eyes.

I am truly grateful for everything all of you have done for me. Without any of you, the completion of this PhD would have been impossible.

Contents

1	Introduction	1
1.1	Aims and objectives	3
1.2	Contributions	4
1.2.1	Publications	5
1.2.2	Others	6
1.3	Thesis structure	6
2	Deep Computer Vision Meets Neuroevolution	8
2.1	Deep Learning	8
2.2	Computer Vision	12
2.3	Multi-Objective Optimisation	18
2.3.1	Evolutionary Computation	21
2.3.2	Evolutionary Multi-Objective Optimisation	23
2.4	Automated Machine Learning	23
2.4.1	Neural architecture Search	24
2.4.2	Neuroevolution	25
2.4.3	Proxy scores	31
2.5	Applications to Concealed Threat Detection	33
3	MONCAE - Multi-Objective Neuroevolution for Convolutional Autoencoders	38
3.1	Motivation study	40
3.1.1	Rationale	41
3.1.2	Methodology for testing Rationale	42
3.1.3	Experimental setup for testing Rationale	44
3.1.4	Results of testing Rationale	44
3.1.5	Why is MONCAE important?	46
3.2	MONCAE Methodology	47
3.3	MONCAE algorithm	49
3.4	Experimental setup	54
3.5	Results	56
3.6	Conclusion and Future Work	60

4	RAMOSS - Resource-Aware Multi-Objective Semantic Segmentation	62
4.1	Methodology	65
4.1.1	Novel encoding and decoding approach	65
4.1.2	Benchmarking framework	68
4.1.3	Progressive Stratified Sampling (PSS)	70
4.2	Experimental setup	72
4.3	Results and Discussion	76
4.4	Conclusion and Future work	81
5	Applications of RAMOSS and MONCAE to Concealed Weapon Detection	83
5.1	MEOW - Automatic Evolutionary Multi-Objective Concealed Weapon De- tection	85
5.1.1	Related work	87
5.1.2	MEOW Methodology	87
5.1.3	Experimental setup	91
5.1.4	Results and Discussion	91
5.1.5	Conclusion and Future work	102
5.2	MIRA-ME: Multi-objective coReset discovery through evolutionAny algo- rithMs in computEr vision	103
5.2.1	Coreset Discovery	104
5.2.2	Methodology	107
5.2.3	Experimental setup	111
5.2.4	Results and Discussion	112
5.2.5	Conclusion and Future work	118
6	Concluding Remarks and Future Work	119
6.1	Thesis summary and main contributions	119
6.2	Implications	124
6.3	Future work and limitations	126
	Appendices	129
A	X-Ray screening	130
B	Convolutional Neural Networks	132
B.1	Convolutional layer	132
B.2	Pooling layer	135
B.3	Upsampling layer	135
B.4	Hierarcical Convolutional Neural Networks	136
B.4.1	Hyperparameters	136

C Representation Learning	139
D History of computer vision	141
E State-of-the-Art Computer Vision Methods	144
F State-of-the-Art Neural Architecture Search Methods for Computer Vision	152
G Datasets	156
H Anomaly detection	159
I Encoding convolutional neural network architectures	162
J Concealed Weapon Detection using State-of-the-art Object detection	165
K Preference articulation through loss manipulation	167
L Density plots for number of pixels per class over images	171
M Discovered best Cityscapes model visualisation	173

List of Figures

2.1	The process of upsampling with a transposed convolution with a 2x2 kernel and a stride of 1.	10
2.2	General structure of symmetrical convolutional autoencoder	11
2.3	The main characteristics of an approximation set, the dominated objective space and the Pareto front can be observed. The Pareto front in the image is represented by the line drawn between the solutions (black and green dots)	19
2.4	The process followed by most evolutionary algorithms with the major steps presented and an example of breaking steps into sub-steps is presented for Variation . Splitting variation into Crossover and Mutation is usually done for a subset of evolutionary algorithms called Genetic algorithms. . . .	21
2.5	A high-level view of the neuroevolution process.	27
2.6	SIXRay sample with multiple overlapping threats	36
3.1	General structure of symmetrical convolutional autoencoder	41
3.2	The three used architectures in the "motivation experiment". Architecture a) is a simple convolutional autoencoder with only three layers before the encoding, b) has four layers before the encoding as well as more than double the feature maps available to a), and c) is significantly more complicated with having three resnet blocks (equivalent to 15 layers as counted for the other architectures) before the encoding.	43
3.3	Distribution of accuracies for each experimental arm	45
3.4	Box plot of performance across the three experimental arms for motivation test experiment for MONCAE. Notice how the Control arm does not outperform the second experimental arm but seems to outperform the first experimental arm. This finding can be attributed to the fact that both the decoder and the encoder are optimised during training, but the decoder is subsequently discarded.	46
3.5	High-level flowchart of neuroevolution with all steps from general evolutionary algorithms with Decode and Encode operators added. The Variation operator can be further split into Crossover and Mutation in the case of Genetic algorithms.	50

3.6	Decoding process if feature map with odd height or width appears. The green(downscaling) and the yellow (bottleneck) layers are part of the encoding. The red(upscaling) layers are generated based on the shape of the bottleneck and input size.	52
3.7	A population of solutions for MNIST. A single digit is used as an example of the performance of every single model from a run picked randomly. For each model, an example of the reconstruction of a single digit can be seen, as well as the achieved reconstruction loss and level of compression. . . .	56
3.8	MNIST autoencoder with bottleneck of 4x4x4 and reconstruction loss of 0.0817. Notice that the digits are almost indistinguishable from the original inputs despite being reconstructed from a 12 times smaller representation.	59
3.9	Fashion-MNIST autoencoder with bottleneck of 4x4x4 and reconstruction loss of 0.289. Notice that while some of the details are lost due to the high compression in the bottleneck layer, the different clothing articles are still easily recognisable, meaning that the salient information for classification is conserved in the 4x4x4 representation.	59
3.10	CIFAR-10 autoencoder with bottleneck of 8x8x8 and reconstruction loss of 0.564. In difference to the MNIST and Fashion-MNIST results, here the convolutional autoencoder attempts to capture more of the colours than the actual features or concepts of what makes up a class. Speculation can be made that this is because colour can be a "loud concept" (Kazhdan et al. 2021); thus, the loss function needs to consider this.	60
4.1	Neuroevolution process stages. The main contributions are highlighted in the shaded green boxes.	65
4.2	The process of decoding integer parameters for connections to decoded operations.	66
4.3	Mean, standard deviation, maximum and minimum values of pixels of each class compared between two PSS splits and the original dataset. Only two out of the ten splits are shown to enhance legibility.	71
4.4	Mean, standard deviation, maximum and minimum values of pixels of each class compared between several random splits and the original dataset. Only 6 out of 10 splits are shown for legibility.	72
4.5	A sample image from Cityscapes from the city of Zurich, where a) is representative of the inputs for the model and b) presents what the labels look like.	74
4.6	An example of different encoding approaches resulting in duplicate architectures	75

4.7	Convergence of models during training with and without PSS . Blue are the training runs on the full dataset, whereas the red lines are the runs with PSS.	77
4.8	Validation loss for PSS-5 and PSS disabled on Cityscapes with UNet with three different backbone architectures.	77
4.9	RAMOSS results on CIFAR10. Figure 4.9a shows the convergence of RAMOSS on CIFAR 10 for 20 generations run with a population size of 20, and Figure 4.9b illustrates the non-dominated solutions in the objective space found during the same run where f_1 is the loss objective during the search and f_2 is the level of complexity	80
4.10	NASBench Pareto-front of found solutions. Since both f_1 and f_2 are minimisation functions, notice that RAMOSS with NSGA-II(Blue) outperforms random search(Red) by presenting a front of solution with a higher hypervolume.	81
5.1	Sixray class labels. Notice the severe imbalance between the "Negative" class and the rest. Also, the " Scissors " class is the most undersampled class, and the training and testing samples seem to follow a similar distribution.	85
5.2	Sample images from the two datasets	87
5.3	The MEOw algorithm. The population of architectures are encoded using lists of integers (a), and then they are decoded to architectures using RAMOSS (b). Panel b) shows a condensed version of the MEOw-s1 architecture for SIXray. Sub-figure c) presents the activations of a selected layer of the trained MEOw-s1 when a sample input (d) is passed through the network. e) presents a sample objective scores of an architecture in the population, which is used for selection. Finally, f) shows the raw predictions of MEOw-s1 for input d).	88
5.4	Hierarchical refinement process from Miao et al. (2019). a^l signifies the activation a at layer l of the architecture. g is the function from CHR that determines and filters the noise. The $+$ denotes concatenation, and \tilde{a} denotes the filtered activation after g is applied.	94
5.5	MEOw-opt architecture for SIXray (truncated after the last convolution to enhance legibility).	97
5.6	MEOw-opt architecture for the Residuals dataset.	99
5.7	MIRA-ME variation process. Notice that during the variation, the selected coreset encodings reproduce, and then variation is applied for the effective exploration and exploitation of the search space.	107

5.8	MIRA-ME images encoding process. Notice that each image x_i has an associated weight in the genome w and the threshold(w_{n+1}) used to determine if an image is "important enough" to be in the coreset is part of the genome and is thus subject to the discovery process of the evolutionary algorithm.	108
5.9	"Naive" MIRA-ME evaluation process. The two objectives considered here are the number of images and the task error rate. Notice that the green dots (representing non-dominated solutions) are the solutions that are going to be selected during the selection process based on their contributing hypervolume. As part of the evaluation, the network is trained with the selected coreset \hat{x} for each solution in the population (dot in the scatter plot on the left).	109
5.10	Evaluation in the "unsupervised" pipeline. Here there are 3 objectives: number of images, reconstruction loss and distance of latent representations. The latent representations are captures from the trained convolutional autoencoders used for the particular dataset. The autoencoders used here are discovered using MONCAE. Notice that in the sample provided, the threat is clearly visible in the reconstructed image.	110
5.11	MNIST MIRA-ME results. Random, Glister and SimCLR are run with 10% and 1% of the dataset, and Random is run with 0.1%. Notice that in the ultra-low data regime (below 1%), Random is clearly outperformed by genetic algorithms such as EvoCore and MIRA-ME with maximum compression (MIRA-ME (naive) mc). MIRA-ME (naive) mc stands for the coreset of the final population that has achieved the maximum compression. Also, notice that both the naive and unsupervised versions of MIRA-ME outperform Random with significantly fewer samples. While SimCLR and Glister beat them, they use over 1000 more samples (and over 4000 in the case of the unsupervised pipeline) than MIRA-ME.	113
5.12	MNIST pixplots	113
5.13	CIFAR-10 MIRA-ME results. Notice that the advantages of MIRA-ME here are even more prominent. While both pipelines are slightly outperformed by the other approaches when they use more than 2x the data, both MIRA-ME pipelines (using less than 8% of the data) outperform all approaches when they use 10% of the data. Interestingly, the unsupervised pipeline here is again noticeably stronger than the naive pipeline and uses even fewer samples.	114
5.14	CIFAR-10 pixplots	115
5.15	SIXray10 pixplots	116

B.1	An example convolution operation with a kernel(F) of size $2 \times 2(m, n)$ and a stride of $1, 1 (sx, sy)$ iterating over 5 by 5 input (X) and producing 4 by 4 feature map (\hat{X}) where each $\hat{X}_{i,j} = \sum_{m-1}^{h=0} \sum_{n-1}^{w=0} F_{h,w} X_{i*sx+h,j*sy+w}$. So when the stride is $1, 1$ it can be simplified to $\hat{X}_{i,j} = \sum_{m-1}^{h=0} \sum_{n-1}^{w=0} F_{h,w} X_{i+h,j+w}$.	133
B.2	Dilated convolution kernel compared to a normal convolutional layer kernel	134
B.3	The process of upsampling with a transposed convolution with a 2×2 kernel and a stride of 1.	136
B.4	Figure depicting the result of a learning rate finder. The learning rate is plotted together with the corresponding loss throughout 1 epoch of learning rate change. Learning rate is changed every n^{th} training step, where n is subject to the number of learning rates that will be explored and the number of mini-batches of the data. The steepest fall signifies a good initial learning rate.	137
E.1	Inception block used in Szegedy et al. (2015a).	144
E.2	Residual block used in He et al. (2015). The identity of the input is added together with the output of <i>layer 2</i> (in this example before the next activation is done.)	146
E.3	Dense block with 4 layers used in Huang et al. (2017). Each layer outputs its feature map to each following layer in the neural network which has a matching feature-map size.	146
G.1	Example images with all 10 classes of CIFAR10 (taken from https://www.cs.toronto.edu/~protect/unhbox/voidb@x/protect/penalty/@M\{}kriz/cifar.html) (Krizhevsky et al. 2014)	157
I.1	Illustration of the encoding/decoding process. a) presents the same architecture described in b), but a) is decoded and b) is encoded. c) presents a more in-depth view of how the connections array from b) is decoded and d) is an example b) of a connections array which is $[128, 112, 33, 16, 8, 4, 2, 1]$	164
J.1	Object detection experiment results. In the first two figures (a) and (b) the x axis denotes the training steps (how many minibatches the algorithm iterated over) and the y axis denotes the mean average precision in (a) and the loss in (b)	166
L.1	Here each image from (a) to (t) displays the distribution of number of pixels for each class separately over all images in the dataset.	172
M.1	Visualisation of best discovered model with MOSS for Cityscapes	174

List of Tables

3.1	Normality test results for the three experimental arms, where s is the skew test, and k is the kurtosis test. The test operates under the assumption that the null hypothesis is that the data comes from a normal distribution and anything above 0.05 p-value is considered high enough to reject this hypothesis.	45
3.2	MONCAE search space. Notice that these presets can be treated as high-level hyperparameters and can be easily modified to construct new search spaces.	50
3.3	MNIST results from 10 independent runs using a conventional autoencoder with variable bottleneck layer size, a simple convolutional autoencoder, one produced from EvoAA, the best one from MONCAE and the MONCAE average. The + symbol denotes that the method is manual, and that is why there is no score (-) for the total time. Notice that the autoencoder discovered by MONCAE with the highest cHv (MONCAE-s1) achieves the best classification loss and accuracy while compressing the data the most out of the whole selection. MONCAE-average results are produced by running MONCAE 10 separate times, and because MONCAE aims to produce Pareto optimal solutions with a high standard deviation (denoted by \pm).	57
3.4	Fashion-MNIST results from 10 independent runs following the same conventions as the MNIST table (3.3) presented above. The + symbol denotes that the method is manual, and that is why there is no score for the total time. Again, the autoencoder discovered by MONCAE with the highest cHv (MONCAE-s1) achieves the best classification loss and accuracy while compressing the data the most out of the whole selection. Similarly to the MNIST results, the standard deviation across the produced populations from the 10 runs is high.	58

3.5	CIFAR-10 results from 10 independent runs following the same conventions as the MNIST table (3.3) presented above. The + symbol denotes that the method is manual, and that is why there is no score for the total time. Notice that this time EvoAA’s autoencoder offers the best compression but is severely outperformed by the other baselines as well as the MONCAE models, which again manage to achieve the best classification accuracy.	59
4.1	Layer parameters defining the search space. n_c , n_d are the number of convolutional and dense layers, respectively (see main text).	66
4.2	CIFAR10 results. RAMOSS runs for 400 model evaluations (20 generations with a population size of 20) with reference points set to (25, 10, 10) for the evaluation of solutions (Dimanov et al. 2021), see details in the main text. † is used to denote manual architectures; hence, their cHV is undefined ”-”. The 0.33 GPU days for search operations exclude the re-training of the network for 300 epochs, which took 2 additional Nvidia 3090 RTX GPU hours. RAMOSS achieves an excellent cost-error-parameters balance (bold font: best values).	78
4.3	Cityscapes results. RAMOSS uses 40,50 and 15 as reference points. ”*” indicates multiple backbone architectures and ”-” for cHV means the solution is fully dominated by another one and does not contribute to the overall hypervolume. Similar to Table 4.2, this table presents the trade-off between the objectives using the cHV (see Table 4.2 for further details).	79
5.1	SIXray-10 results. Notice that the MEOw-opt and MEOw-ens methods achieve the highest mAP, and the state-of-the-art lags behind significantly. Interestingly, for the most oversampled class, ”Pliers” (except ”Negative”) DenseNet is achieving better performance, but the MEOw architectures balance the rest of the classes more effectively and are especially good at classifying the most underrepresented class ”Scissors”.	92
5.2	Residuals results. The residual problem consists of two separate tasks: detecting the type of modification and the threat of the modification. The following table displays the averaged results for both of these serape multi-class outputs and even though marginally, MEOw architectures outperform the state-of-the-art. Remember that the MEOw architectures are substantially smaller than the other ones.	92

5.3	Residuals results in % f-1 score per class. Here, the detailed per-class scores of Table 5.2 reveal that MEOW architectures, consistently with the SIXray results, achieve the best balance of different classes and are outperformed only for one type of modification by a state-of-the-art model with less than 0.5%. Note that MEOW architectures are substantially smaller than the other ones.	93
5.4	Results on Residuals dataset with segmentation masks. In each column the top 3 approaches are highlighted (first, second and third best). While MEOW-opt does not achieve the best test <i>IOU</i> and <i>F1</i> , it remains the only architecture capable of yielding a validation performance which can resemble its performance on an unseen dataset. Another property making the MEOW architecture potentially the most favourable out of the mix is that it uses only 0.53M parameters, compared to 62 and 30 for the other better-performing alternatives.	96
5.5	SIXRAY 10 MIRA-ME results. Notice that the MIRA-ME approaches are better than both Glister and Random, even when these approaches use up to 3x more data. What is more, the 62% achieved by the unsupervised pipeline (using less than 5% of the data) is close to the model's performance with the full dataset of around 77% presented at the beginning of the chapter.	116
K.1	Precision for 0.5 threshold in % for each class and macro mean precision for SIXray 10,100,1000	170
K.2	Average precision in % for each class and macro mean recall for SIXray 10,100,1000 for 15 runs.	170

Acronyms

AutoML Automated Machine Learning.

CNN Convolutional Neural Network.

GAN Generative Adversarial Networks.

HOG Histogram of Oriented Gradients.

MEOW Automatic Evolutionary Multi-Objective Concealed Weapon Detection.

MIRA-ME Multi-objective coReset discovery through evolutionary algorithms in computer vision.

MLP Multilayer Perceptrons.

MONCAE Multi-Objective Neuroevolution for Convolutional AutoEncoders.

NAS Neural Architecture Search.

NASWOT Neural Architecture Search Without Training.

NN Neural Network.

RAMOSS Resource-Aware Multi-Objective Semantic Segmentation.

SYNFLOW Iterative Synaptic Flow Pruning.

XAI Explainable Artificial Intelligence.

Chapter 1

Introduction

Computer vision algorithms have always enjoyed tremendous attention in the machine learning community and have been in high demand in general ever since Sussman's successful Summer project (Papert 1966), through the birth of convolutional neural networks with the Neocognitron (Fukushima and Miyake 1982) and LeCun's successful LeNet LeCun et al. (1998).

Despite presented almost 10 years ago, InceptionNet(Szegedy et al. 2015b) and ResNet(He et al. 2015) remain some of the most popularly used architectures to date (Miao et al. 2019, Wu et al. 2019b) . The problem is that these architectures, like many others, were designed for specific tasks and while achieving good results in plenty of different settings, they are far from optimal solutions in many cases (Miao et al. 2019) and it is not advisable to be taken as the *silver bullet* of computer vision (Guo et al. 2019a, Castilla et al. 2022).

On the other hand, hiring a team of data science engineers to tailor networks for each new industrial application is unfeasible. Hence, researchers have started exploring the possibility of using automated systems that, given an objective and data, can generate fully-fledged models capable of achieving competitive performance with systems designed by experts (Liu et al. 2018, Real et al. 2017, Tan and Le 2019a, Qin and Wang 2019).

While these automated machine learning (AutoML) methods represent the new state-of-the-art in computer vision, they do not come without their caveats. First and foremost, the current state-of-the-art methods are extremely slow, even for low-dimensional datasets- taking thousands of GPU days to discover a good architecture (e.g. 3150 GPU days (Real et al. 2017) and 9000 GPU days (Qin and Wang 2019))(Real et al. 2018, Qin and Wang 2019). The vast majority also focus on a single, performance-related objective, which negatively affects the models' size and the needed run time for sensible results. Moreover, they are mainly used for small-scale, low-dimensional problems and are designed in a way that limits their scalability to larger or simply different datasets (Stanley et al. 2019a).

The current AutoML state-of-the-art primarily focuses on simple binary or multi-class classification tasks (Stanley et al. 2019a). However, the rapidly-evolving field of machine learning research requires AutoML solutions that are more representative of real-world applications and adaptable to various modifications. This advancement is necessary to enable researchers to quickly explore different ideas without having to build new methods from scratch. This gap can be addressed by proposing a novel AutoML approach that is more flexible and suitable for a broader range of applications. Hence, in Chapter 5.1, the plasticity of the presented approaches is demonstrated by restructuring the neuroevolution algorithm from Chapter 4 to operate on the data scape and discover critical data points in a dataset instead of convolutional architectures.

A considerable portion of the AutoML approaches is highly specialised at a specific task (which is usually a benchmarking dataset) instead of being a general solution which they should be by design. While such attempts are a prerequisite for the success of AutoML since they provide much-needed findings about the feasibility of particular approaches, the potential for industrial applications of AutoML remains largely unexplored and has recently attracted the attention of, e.g., Google (Bisong 2019) or Alibaba (Li et al. 2021) among many other corporations.

Among the myriad potential applications of AutoML, security and, more specifically, concealed weapon detection is of particular interest in this thesis. More specifically, one of the main goals of this work is to develop an automated approach for X-ray computer vision in concealed weapon detection to be applied in a range of industrial applications. The efforts towards this goal are presented in Chapter 5.1 where the proposed efficient methods from Chapters 3 and 4 are applied to concealed threat identification, and the discovered convolutional architectures achieve state-of-the-art results in two different such datasets. Interestingly, some of the work done towards this thesis has been used by the award-winning project.

Concealed weapon detection is usually achieved through either millimeter-waves (Rostami 2014), radar (Goenka and Sitara 2022) or X-ray scanners (Miao et al. 2019). Out of these, only X-ray detection is within the scope of this thesis and is discussed in more detail in Appendix A.

In essence, imagery data from X-rays differ significantly from the one coming from visual sources such as CCTV and present novel challenges that need to be addressed with specialised solutions (Miao et al. 2019, Mery 2015). For example, items in close-packed bag X-ray scans are typically difficult to detect for many reasons, including the fact that stacked items are not fully occluded but rather overlaid (Mery 2015). Moreover, X-ray imagery captures the density of a certain object rather than just its exterior (Henzler et al. 2018). This property causes many conventional methods and models to fail in these tasks and showcases the necessity for domain-specific model architectures.

Moreover, most approaches in this area are manual, and the rest phrase the problem

as a binary classification (they are concerned with only detecting if there is a threat or not (Mery 2015, Dimanov 2019)). This method is sub-optimal since different threats need to be handled differently (Bolz Jr et al. 2016). For example, security should approach a person of interest suspected of carrying a firearm and one of having a bomb in two distinctive ways. In many cases, multiple conflicting objectives need to be considered. For example, the performance for a specific class might be more important than the overall performance, or computational constraints (e.g. size or speed of the model) need to be considered.

Concealed weapon detection is just an example of an area in need of simultaneous optimisation of multiple, often conflicting, objectives. More broadly, multi-objective optimisation has a wide range of applications in finance and economics (Tapia and Coello 2007) (e.g., in investment portfolio optimisation, stock ranking, stock-return prediction or in any other form of economic modelling), decision-making in design and engineering Domingo-Perez et al. (2016), Pillana et al. (2019), energy forecasting (Tomoiagă et al. 2013, Liu et al. 2020a), and in many other areas (Ellefsen et al. 2017, Björnson et al. 2013, Ogbolumani and Nwulu 2021, Bagheri-Esfah et al. 2020).

In short, multi-objective optimisation attempts to improve the balanced performance of the chosen method over several different objectives (represented by different objective scores) (Deb 2014), as discussed in further detail in Chapter 2. The need for multi-objective optimisation for AutoML may seem obvious (Stanley et al. 2019a), yet some of the early attempts to achieve this (Real et al. 2018) have been replaced by single-objective reinforcement learning approaches (Tan and Le 2019a). One of the hypotheses of this thesis is that the potential of multi-objective evolutionary algorithms for automated machine learning has not been fully harnessed yet.

Moreover, with the rise in popularity of representation learning and explainability, there has been an increasing interest in unsupervised and manifold learning methods in general (Bengio et al. 2013, Dimanov 2021, Venkataramanan et al. 2022). Consistent with this trend, this thesis contributes to bridging the two fields by proposing a way to conduct unsupervised machine learning in an automated fashion.

1.1 Aims and objectives

This thesis aims to provide a stepping stone towards the widespread use of AutoML in research and industry, emphasising case studies in threat identification. The novel methods, designed to be applicable to various computer vision problems, should automatically identify potential threats in X-ray screening with high enough speed and performance to constitute an effective prevention approach.

Moreover, this work aims to accelerate AutoML while preserving (or improving) its effectiveness. In short, the proposed approaches can boost AutoML availability and its

growth as a research field while reducing the carbon footprint of these highly computationally demanding deep learning methods. Overall, the societal benefits derived from a more efficient AutoML can contribute to broadening the scope of the next generation of AI approaches (Doke and Gaikwad 2021). Towards this goal, the **objectives** of this thesis are to:

1. Identify key strengths and limitations of the state-of-the-art in AutoML for computer vision, with an emphasis on neuroevolution and threat detection.
2. Design an efficient AutoML approach for computer vision, capable of simultaneously optimising multiple objectives (image reconstruction, dimensionality reduction, model complexity) to discover convolutional autoencoder architectures.
3. Develop a novel multi-objective optimisation AutoML strategy of discovering flexibly connected convolutional networks for semantic segmentation which can dynamically adjust to computational requirements.
4. Adapt the newly developed method to real-world dataset using concealed threat detection problem as a case study to showcase potential field applications.
5. Develop a heuristic approach to bolster the feasibility and efficiency of AutoML in both model and data optimisation, facilitating its broader adoption in industrial applications, exemplified through a case study in concealed threat detection.

1.2 Contributions

1. Aligning with Objective 1, an extensive literature review is presented in Chapter 2 that identifies the key strengths and weaknesses of the current state-of-the-art approaches in computer vision, AutoML and multi-objective optimisation.
2. Addressing the challenges highlighted in Objective 2, A multi-objective neuroevolution neural architecture search method for convolutional autoencoders (MONCAE) is presented in Chapter 3. The approach, also presented at an ICLR 2021 workshop, has the potential to impact a wide range of applications due to the increasing popularity of autoencoder-based generative methods.
3. In accordance with Objective 3, a resource-aware multi-objective neuroevolution approach for semantic segmentation - RAMOSS is designed, developed and evaluated in Chapter 4. The encoding of RAMOSS allows for flexible connections between different layers by giving the ability of each layer to connect to each other layer. The speed-up achieved by the approach makes AutoML feasible for segmentation problems and demonstrates the value of multi-objective optimisation. The

approach was presented at the UKCI 2022 conference and has been awarded a prize - Best Presentation in UKCI 2022.

4. Fulfilling Objective 3, a new strategy called Progressive Stratified Sampling - PSS was devised in Chapter 4, which aims to provide a way to sample segmentation and multi-label datasets in a stratified fashion. The strategies allow for substantial acceleration in the training of networks while conserving their performance ranking.
5. In line with Objective 4 and partly Objective 5, the application of Neuroevolution methods to threat identification problems is explored in Chapter 5.1. Through some added heuristics, an automatic evolutionary multi-objective concealed weapon detection (MEOW) is devised that can discover architectures better than the state-of-the-art in a single GPU hour.
6. As a result of pursuing Objective 5 and partly Objective 4, Multi-objective evolutionary coreset discovery technique is developed that allows for a fully-unsupervised search of important samples in a dataset. The approach discussed in Chapter 5.1 (Section 5.2) presents a unique opportunity to discover a small subset of samples from a dataset that a given model can train on and achieve similar performance to the same model training on the whole dataset. With big datasets, this approach can unveil some out-of-distribution or mislabelled samples and implicitly inform the user of the "important patterns" in the dataset.

1.2.1 Publications

The following publications are a result of this work:

1. D.Dimanov, S. Rostami, C.Singleton and E. Balaguer-Ballester - "MONCAE: Multi-Objective Neuroevolution for Convolutional Autoencoders"(Dimanov et al. 2021) [*Published* in the ICLR 2021 workshop for AutoML which became a stand-alone conference in 2022]
2. D.Dimanov, S. Rostami, C.Singleton and E. Balaguer-Ballester - "RAMOSS - Resource-Aware Multi-Objective neuroevolution for Semantic Segmentation" (Dimanov et al. 2022) [*Published* in UKCI 2022 conference] - Awarded Best Presentation in UKCI 2022
3. D.Dimanov, S. Rostami, C.Singleton and E. Balaguer-Ballester- "MEOW: - Automatic Evolutionary Multi-Objective Concealed Weapon Detection"[*Published* in AutoML 2023]
4. D.Dimanov, S. Rostami, C.Singleton and E. Balaguer-Ballester- "MIRA-ME: Multi-objective coReset discovery through evolutionary algorithms in computer vision" [*In preparation* for ICML 2024]

5. S.Rostami, A.Kleszcz, D. Dimanov, V. Katos "A Machine Learning Approach to Dataset Imputation for Software Vulnerabilities" (Rostami et al. 2020)[*Published in MCSS 2020*]

1.2.2 Others

1. Delivered a lecture for MSc students on Neuroevolution
2. Presented "MONCAE: Multi-Objective Neuroevolution for Convolutional Autoencoders" at BU PGR conference 2021
3. Presented early results of "Neuroevolution applied for concealed weapon detection" at BU PGR Sci-tech conference 2020
4. Presented a poster for "MIRA-ME: Multi-objective coReset discovery through evolutionary algorithmMs in computer vision" in the BU PGR conference 2022

1.3 Thesis structure

The rest of the thesis is comprised of five themed chapters.

Chapter 2 progressively introduces some core concepts in deep learning, computer vision and multi-objective optimisation. First, the inner workings of neural networks, convolutional neural networks, and autoencoders are discussed. Then, the chapter sets out to establish the state-of-the-art as well as popular datasets in computer vision. Next, multi-objective optimisation is explained with an emphasis on evolutionary algorithms, Pareto-front of solutions and the hypervolume indicator. What follows is a bridge between all three topics discussed so far with the idea of AutoML in computer vision and presents some state-of-the-art models in the field while reviewing their main limitations, such as the enormous computational requirements, lack of experiments on different problems and datasets, single-objective centrism and unfeasibility to be used for real-world applications. One such area for application described lastly in the chapter is concealed threat detection with a specific focus on X-ray imagery data.

Following the discovered research gaps in Chapter 2, Chapter 3 presents a novel neuroevolution approach for discovering convolutional autoencoder architectures named MONCAE. It starts with a motivation study highlighting the importance and potential applications of the discovered convolutional autoencoders from MONCAE. This study demonstrates that autoencoders can be fine-tuned to perform supervised tasks more effectively by leveraging the learned representations in their encoder layers after being trained in an unsupervised manner. Then, the chapter presents the automated way of discovering convolutional autoencoders, which efficiently achieves the best balance between bottleneck compression and classification performance on several different datasets.

Using the findings and the methodology of MONCAE as an inspiration, Chapter 4 addresses some of the limitations of MONCAE (the lack of layer connections in the genome encoding and incompatibility with segmentation problems) by introducing a novel resource-aware multi-objective neuroevolution-based semantic segmentation approach in the face of RAMOSS. RAMOSS is evaluated using a popular segmentation dataset-Cityscapes- and complementary experiments with a prevalent classification task for benchmarking AutoML methods (CIFAR-10). On both problems, the RAMOSS architectures achieve the best balance between the number of parameters, performance and search time. Together with RAMOSS, chapter 4 also proposes a new stratified sampling method compatible with multi-label and segmentation datasets. The strategy is aimed towards automated machine learning approaches and plays a vital role in the efficiency of RAMOSS.

Chapter 5 seeks to assess the feasibility of using the presented approaches in Chapter 3 and Chapter 4 for real-world concealed threat identification. In line with the rest of this work, the chapter introduces some heuristic scores, which are then utilised to speed up the approaches further. This improved version of the neural architecture search methods is then employed to two concealed threat identification datasets (SIXray and a proprietary dataset called "Residuals") and beats all state-of-the-art approaches used for comparison. The chapter continues by exploring the idea of porting the model-space contributions (discussed in the previous chapter) and creating an evolutionary-based critical dataset discovery method named MIRA-ME. MIRA-ME has two different versions - "naive" and fully "unsupervised"- both demonstrate promising results on standard computer vision and domain-specific datasets.

The final chapter (Chapter 6), reflects on the extent to which the objectives outlined in the current chapter have been met through the research presented in this thesis, including a discussion of the main findings and contributions of the work, as well as the implications of the results and recommendations for future research.

Chapter 2

Deep Computer Vision Meets Neuroevolution

This chapter describes the main concepts, techniques and open problems discussed in the thesis. It begins with a brief introduction to the field of deep learning. Since the problems addressed in this work are all in the remit of computer vision, an overall view of the field is presented next. As presaged by the Introduction, a core focus of this thesis is multi-objective optimisation. Hence, the chapter proceeds to discuss the field in conjunction with evolutionary computation. Finally, the niche topic of this thesis, automated machine learning (AutoML), is introduced, which presents a bridge between deep learning and multi-objective optimisation.

2.1 Deep Learning

To contextualise **deep learning**, it is convenient to start more broadly by discussing **machine learning**, which dates back to at least 18th century (Bayes 1763). Mitchell (2006) describes the field of machine learning as the approach which attempts to answer an essential question: "How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?".

It is a widely held view that the main characteristic of machine learning is that knowledge is acquired from experience rather than being specified explicitly by humans (Murphy 2012, Goodfellow et al. 2016b). Moreover, it provides a way of conducting automated methods for data analysis (Murphy 2012).

As machine learning continues to evolve, it may be argued that the natural next step is the adoption of AutoML (Stanley et al. 2019a). This approach to model generation and optimisation allows for the further automation of the knowledge extraction process. By leveraging experience-based algorithms, AutoML represents a logical progression in the field, moving us closer towards a fully autonomous approach to machine learning (He

et al. 2021, Li et al. 2021).

Due to the enormous size of this research field, only a small part of it, which is integral for the complete understanding of this thesis, is reviewed. More information is provided in Appendices B and B.4.1.

Neural Networks: Neural networks are biologically inspired representations of how many scientists believe our brains work (Goodfellow et al. 2016b) and while it can be argued that the field is a multi-disciplinary symbiosis, the working principle of these networks is mainly based on linear algebra (Cochocki and Unbehauen 1993), calculus (Rashid 2016) and statistics (Weiss and Kulikowski 1991).

Feedforward neural networks are typically represented by composite functions, which are arranged in a directed acyclic graph (DAG) and represent a chain of operations that needs to be applied to the input (Goodfellow et al. 2016b). DAG is also what is used as part of the decoding in all of the following core chapters. For example, Equation 2.1 illustrates a composite function g of five nested functions (f_1 through f_5), where the input is passed to f_1 (which is often referred to as the first layer), then to f_2 (second layer), until it reaches f_5 , which is also the last layer (Goodfellow et al. 2016b).

$$g(x) = f_5(f_4(f_3(f_2(f_1(\mathbf{x})))))) \quad (2.1)$$

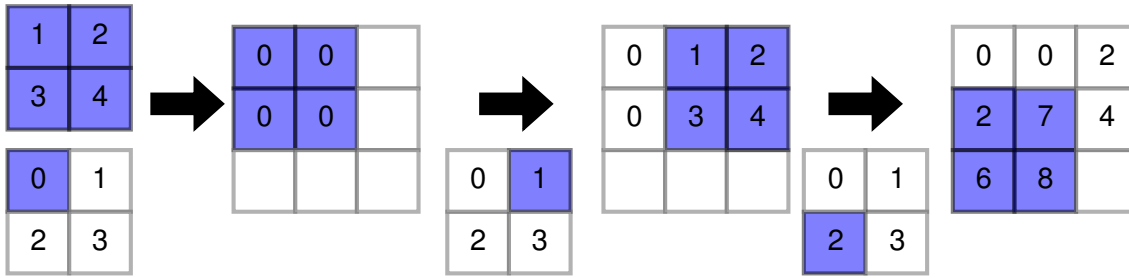
The depth of the network is determined by the number of layers, which also means the number of functions in the chain from the example above (Goodfellow et al. 2016b). In the provided example, f_1 would also be called the input layer, and f_5 would be the output layer. All of the layers represented by f_2 to f_4 are called hidden layers.

Convolutional Neural Networks: The unique properties of Convolutional Neural Networks (CNNs) and the fact that convolutional as well as pooling layers use kernels instead of fully dense connections allows them to be especially good at learning representations with multiple levels of abstraction (Yang et al. 2017a) and effectively process high-dimensional and complex data (Pereira et al. 2009).

Aside from traditional convolutional and pooling layers, CNNs can use other operations to complete data-specific operations, such as dilated convolutions (Hamaguchi et al. 2018), which have benefited segmentation models.

In contrast, the **upsampling layer** aims to reverse the operation of the pooling layer (Goodfellow et al. 2016a). Upsampling is usually done to restore the image's original dimension, which is an integral part of convolutional autoencoders and segmentation models (Ronneberger et al. 2015a). The upsampling layer uses a kernel similar to the pooling layer, but instead of mapping to a lower-dimensional representation space, it maps to a higher-dimensional one. There are two general ways to achieve this in a

Figure 2.1: The process of upsampling with a transposed convolution with a 2x2 kernel and a stride of 1.



convolutional neural network. The first one uses an upsampling layer, which scales up a data representation using the nearest neighbour or bilinear upsampling.

The way representations are decompressed is through the use of a transposed convolution, the inverse operation of a strided convolution. Transposed convolutions came to light as a means to upsample features using a kernel with weights (Zeiler and Fergus 2014). These are also termed 'deconvolutions' or 'fractionally strided convolutions'. Figure 2.1 presents an example of a transposed convolution. The figure depicts the iterative process of the kernel (top left 2x2 grid) going over the input (the bottom 2x2 grid). Each cell from the input is multiplied by the kernel at each step. Then, all produced feature maps are added using a specified step size (stride). In the example, the stride is 1, so each cell where there is overlap represents the sum of all produced overlapping cells.

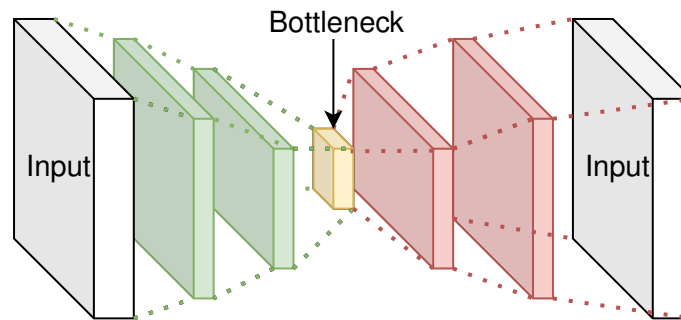
More information and a detailed discussion of the inner workings of CNNs and their operations can be found in Appendix B.

Autoencoders: One of the contributions of this thesis and the focus of Chapter 3 is the automatic discovery of convolutional autoencoders. Autoencoders are neural networks that have symmetric input and output layers and usually contain a bottleneck layer (as illustrated in Figure 2.2) (Schmidhuber 2015). Usually, they are trained to minimise the reconstruction loss, which is calculated based on the difference between the original and the reconstructed (output) ones.

The general working principle of these networks is that they "are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion." (Baldi 2012). The fact that this approach utilises the data in an unsupervised way allows for various applications ranging from style transfer (Qian et al. 2019) and generative techniques (Dosovitskiy and Brox 2016) to semi-supervised learning (Akçay et al. 2018), data denoising (Gondara 2016) and image compression techniques (Charte et al. 2020) that utilise the intrinsic representation learning of these approaches and exploit it.

The simplest form of an autoencoder is a three-layered MLP with a single hidden layer, which represents the encoding, and the difference between the output and the input is

Figure 2.2: General structure of symmetrical convolutional autoencoder



then calculated, which serves as a reconstruction loss (Sereno 2018). Usually, autoencoders are made up of two parts: the encoder, which is the subnetwork that condenses the input to a small representation (chokepoint) and the decoder, which is the rest of the network, which uses this representation to upscale and reproduce the input (Goodfellow et al. 2016b). For sophisticated datasets, convolutional layers can substitute the simple dense ones. These autoencoders are referred to as convolutional autoencoders (Azarang et al. 2019). The difference with simple autoencoders is that they achieve compression through convolutional and pooling layers instead of simply having fewer neurons. The upsampling is achieved through deconvolutions (Baldi 2012) discussed above. Autoencoders are powerful representational learning models, but some aspects, such as their explainability and required representational capacity, remain unsolved, as discussed in Appendix C.

The state-of-the-art in autoencoders is composed of architectures designed by experts, which are optimised for the specific scenarios required (Charte et al. 2020). Currently, given the increasing popularity of Automated machine learning (AutoML) (He et al. 2021, Real et al. 2019, Lu et al. 2019, Yan et al. 2020), approaches for automatically designing and optimising autoencoder architectures are starting to appear (Charte et al. 2020), as will be further discussed in Chapter 3.

Model complexity: In Chapter 3, an operational measure of model complexity specifically designed for deep convolutional models is defined, given that

there is generally no established consensus on the way to quantify the complexity of a deep architecture (Lorena et al. 2019). To estimate it works in the field, use the computational complexity (including the size of the model in terms of layers and parameters) (Lorena et al. 2019, Real et al. 2018), representational complexity (Goodfellow et al. 2016b) or how fast an algorithm is at runtime (Lu et al. 2019). It is crucial to determine the complexity of models not only because specific scenarios require rapid predictions (Morris et al. 2018, Miao et al. 2019), but also because the complexity of a model is proven to be linked to the representational or classification capacity of a given model (Goodfellow et al. 2016b)

A classic approach consists of exploring the geometrical complexity of a dataset by

embedding it in a constructed measurement space, which in turn "can guide the static and dynamic selection of classifiers for specific problems as well as subproblems formed by confinement, projection, and transformations of the feature vectors" Ho and Basu (2002). This approach raises awareness of the critical importance of exploring classifiers' dependency on data and a way to devise a metric that measures data complexity. It also reveals a significant gap in the literature: while good and accurate models are presented each year, a small part of the papers, if any, focus on the actual reasons why the chosen method was successful (Arrieta et al. 2020). This open problem is also one of the central questions in the rapidly growing field of XAI (explainable AI), which has gained traction over the last years (Došilović et al. 2018, Gunning et al. 2019, Dimanov 2021).

Works like the Lottery-ticket hypothesis (Frankle and Carbin 2018) and Kazhdan et al. (2020) present heuristics and methods to better understand the black-box nature of deep learning models and as the author of Dimanov (2021) has put it: "To shine a light and observe the black-boxes from the inside". Thus, one of the central themes in the new field of XAI is an evolution of the efforts to estimate models' capacity and complexity and the findings of 20-year-old studies "that many classification tasks arising naturally from real-life processes do contain learnable structures" (Ho and Basu 2002) are currently reincarnated in XAI as "concepts" (Kazhdan et al. 2020).

2.2 Computer Vision

The idea to make computers able to perceive the world like the human eye has sparked interest for decades (Huang 1996). Although the field of computer vision is interdisciplinary and encompasses a multitude of integrally different approaches, the term describes systems that are capable of achieving some high-level understanding of digital images or videos (Huang 1996, Forsyth and Ponce 2002).

Computer vision is a branch of machine learning that deals with translating digital input data (in the form of images and videos) into valuable information (Lawrence et al. 1997) (Dimanov and Rostami 2019). There are different types of computer vision problems (image classification (Khan et al. 2018) , object detection (Cyganek 2013) , semantic segmentation (Long et al. 2015)) and numerous methods that address them (Szeliski 2011, Borkowski et al. 2019).

The field is composed of two main parts. The first one is feature engineering, which includes feature extraction, feature selection and various preprocessing techniques (Szeliski 2010). The second one depends on the exact problem at hand, but in essence, it consists of using the discovered features to solve a particular problem. For instance, in image classification, the discovered features are associated with a set of classes through classifier models (Varano 2017).

At first, the field relied on hand-crafted features, including HAAR-like (Viola and Jones

2004) and HOG (Dalal and Triggs 2005) ones. With the work of Krizhevsky et al. (2012), this changed, and feature extraction together with feature classification was delegated to CNNs (Lang et al. 1990, LeCun et al. 1998). This new revolutionary, yet old, idea could work much better than its predecessor because of the vast increase in computational power and available data in 2012 compared to the 1960s (Alom et al. 2018). Please check Appendix D for more details.

Significant advancements: What follows is a short description of some of the central related works used throughout this thesis. A more verbose and detailed version with additional approaches can be found in Appendix E. All of the following discussed methods use unique architectural and overall design decisions, which is encoded as part of the design of the presented search space in the following chapters. One of the first and most popular architectures for computer vision to date is the **VGG** method Simonyan and Zisserman (2014). The approach was initially designed for a fixed size 224x224 RGB images, which are passed to a series of convolutional layers with only the smallest possible receptive fields that can capture direction (3x3) (Simonyan and Zisserman 2014). Then, some of the convolutional layers are followed by max-pooling layers with a 2x2 kernel and a stride of 2. The output is then flattened and passed to two fully connected layers; the first has 4096 channels, and the last one has 1000 channels, representing the 1000 classes of ImageNET. Multiple models exist based on the VGG architecture based on the number of layers (Liu and Deng 2015). Some of the most popular ones are VGG-16 and VGG-19. The idea of flattening the features of the last convolution and then putting auxiliary classification layers has how the classification version of RAMOSS in chapters 4 and 5 work.

Between 2014 and 2016 **InceptionNet** (Szegedy et al. 2015a) and **ResNet** He et al. (2015), which featured skip-connections that allowed for a better flow not only of information during a forward pass but also proved to have a positive impact on the flow of gradients during a backpropagation. Even though multiple variations exist of both base architectures (Jin et al. 2016, Szegedy et al. 2016a 2017a), the main working principle is that instead of following the previously standard sequential stack of layers, these architectures skip and aggregate over multiple operations. Soon after, an architecture with all layers interconnected called **DenseNet** was proposed Huang et al. (2017). Unlike ResNet, DenseNet concatenated the residual feature maps instead of summing them. One of the decision variables in the encoding in RAMOSS is exactly which operation to use with skip connections, and the available choices stem from these architectures.

Focusing on segmentation, one of the most popular family of models is **U-Net** Ronneberger et al. (2015b). This group of architectures use two mirrored pathways for first "capturing context" and then expansion to facilitate precise localisation. It is based on how autoencoders work, explained earlier, with the only difference being that layers in the encoder have skip connections to the ones with the exact dimensions in the decoder.

As computer vision encompasses so many different problems, approaches and techniques, it is challenging to generalise all concepts as applicable to each problem. Yet, an immense amount of transferable knowledge can be applied from one computer vision problem to another (Torrey and Shavlik 2010). One concept that exploits this postulate is the use of **backbones**, which are neural network architectures or parts of architectures, which are taken and used as part of some larger pipeline (Ren et al. 2015). An example of such an application is the use of ResNet (He et al. 2015) backbones in the construction of UNets (Huang et al. 2020) or the use of InceptionNet (Szegedy et al. 2015b) and ResNet (He et al. 2015) as part of object detection, where they are only used to classify the object in a region, which is separately discovered by a regression model (in the case of R-CNN with such backbones) (Li et al. 2018) . More information about these models and architectural designs can be found in Appendix E.

¹ Over the last decade, deep learning has become a flourishing field, capable of providing solutions to problems that were previously arduous or unforeseeable (Goodfellow et al. 2016b, Li et al. 2023). Deep learning typically scales well with the abundance of available data (Goodfellow et al. 2016b); however, there are scenarios where the data is of overwhelming quantity and training an algorithm on the whole raw dataset becomes infeasible (Miao et al. 2019).

In these cases, two approaches are usually undertaken. They are either based on reducing the number of samples by selecting the most representative ones (Triguero et al. 2015, Jankowski and Grochowski 2004) or reducing the dimensionality of the dataset (Charte et al. 2020), which is especially prevalent in fields like computer vision.

As CNNs have achieved exceptional results and have become the *"golden standard"* for most computer vision problems (Shen et al. 2023, Wu et al. 2023, Simonyan and Zisserman 2014, He et al. 2015, Szegedy et al. 2015b) . CNNs offer a black-box solution that addresses both feature engineering and the classification itself (Liu et al. 2023, Zhang et al. 2023b) . In this setting, it is common to rely on the universal approximation theorem (Hornik et al. 1989) to find a model capable of reasonably capturing the input data distribution conditioned to each class label. If that is not the case, the natural, heuristic approach is to look for deeper models and intuitively experiment with hyperparameters until satisfactory performance is achieved (Goodfellow et al. 2016b).

CNNs can successfully map complex and obscure input/output patterns (Deng et al. 2023b) allowing for new techniques such as attention mechanisms to emerge Vaswani et al. (2017). Moreover, in contrast to earlier methods where features had to be manually specified (ex., HAAR-like features), they automatically discover such optimal features through their optimisers that sometimes carry extra semantic value Jha et al. (2023).

Even though CNNs are potent tools for computer vision problems (Aggarwal et al.

¹VIVA change comment: Moved from MEOW-related works with some updated references and added new techniques

2021), constructing them involves several critical design decisions that can significantly affect the performance of the network (Real et al. 2017). These decisions include choosing the right architecture for the network, determining the appropriate size and stride for the convolutional filters, and selecting the proper activation functions and pooling layers, which all need to be optimised in order to achieve optimal results. Until recently, researchers relied mainly on trial and error intuition to determine the best hyperparameters and network architecture (Stanley et al. 2019a).

However, discovering the state-of-the-art itself often relies on approaches like brute-force grid search or random search (Liashchynskiy and Liashchynskiy 2019, Nałçakan and Ensari 2018, Liashchynskiy and Liashchynskiy 2019).

To address this, recently, there has been an increase in effective approaches for automated convolutional network design, such as evolutionary algorithms (Lu et al. 2019, Miikkulainen et al. 2019, Real et al. 2019, Stanley et al. 2019a), reinforcement learning (Qin and Wang 2019, Tan and Le 2019b) and other automated approaches (He et al. 2021). These methods have been able to match and sometimes surpass manually designed architectures by domain experts and are discussed in further detail in Section 2.4 and then utilised in Chapters 5.1, 4 and 5.

Reducing the computational complexity of these algorithms is a significant research challenge (Zhou et al. 2020). A promising approach is to leverage dimensionality reduction techniques to simplify the underlying data and enable more efficient processing. Interestingly, Charte et al. (2020) discovered that "choosing a curated set of attributes, or building a new one from the original features, tends to produce better results than using raw variables". Standard statistical choices for dimensionality reduction in the literature include Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA) and Factor Analysis. However, more advanced work explores the possibility that the distribution of variables in the original dataset lies in lower-dimensional space (also known as a manifold (Talwalkar et al. 2008)). Manifold learning is used to compress the raw data (using non-linear transformations) to its lower-dimensional representations that essentially contain all the necessary information to complete the task at hand successfully (Charte et al. 2020). This concept has served as key motivation behind Chapter 5.1 and the MIRA-ME technique described in Chapter 5.2. These new techniques of manifold learning have motivated recently reemerging vector databases (Girdhar et al. 2023), which in combination with embedding models (Barkan et al. 2020) offer a unique new way of accessing and searching for dataset-level operations (Rodriguez et al. 2023). This allows users to quickly find similar items in a database or develop context-specific queries that are key for managing machine learning datasets at scale (Babenko and Lempitsky 2016, Deng et al. 2023a).

Typical problems: Having discussed "how" different techniques work, now some of the core problems in computer vision are introduced. **Image classification** is one of the

most mature and fundamental computer vision problems (Nath et al. 2014). It is usually associated with a problem where a single image should be classified to belong to one of two (binary classification (Parkhi et al. 2012)), one of many (multiclass classification (LeCun et al. 1999)) or multiple different (multilabel classification (Miao et al. 2019)). Each specific dataset is introduced as part of the experimental setup of the first chapter it is utilised in.

In difference to pure image classification, where one image is just associated with a label, the **object localisation** task takes this one step further by requiring the system to provide information not only by a single label but rather give context as to where the actual entity is positioned (Tompson et al. 2015). In contrast to object detection, however, object localisation is just concerned with finding one particular entity, thus is more similar to image classification with some auxiliary regression over four parameters which define a single bounding box (Harzallah et al. 2009).

In contrast to image classification, where each image is given a label, in **semantic segmentation** problems, each pixel of the image has its own label (He et al. 2017). Contrary to object detection, where there are multiple defined regions and bounding boxes which can dynamically (Ren et al. 2015) or statically (Redmon et al. 2016) determine whether there is an object in a particular region or object localisation which builds upon image classification with the addition of localising one particular object (Haralick and Shapiro 1985) , semantic segmentation takes this further.

Tasks are based on image masks that give a separate label to each input image pixel. In this particular problem, metrics such as accuracy are not widely used since they are not highly informative (Fenster and Chiu 2006) . Since background classes usually occupy most of the image and other classes are imbalanced, some of the used metrics for this type of problem are intersection over union (Shaban et al. 2017, Liu et al. 2019a) and dice score (Bertels et al. 2019).

Effective semantic segmentation is fundamental for capturing the main characteristics of the scene (Paszke et al. 2016, Chen et al. 2014). For instance, in medical and security domains, semantic segmentation is a crucial part of multiple critical systems, including diagnosis (Taghanaki et al. 2021), health monitoring (Yang et al. 2017b), ensuring people's safety (An et al. 2019) and many others (Alonso et al. 2020, Sagar and Soundrapandiyan 2020). However, the vast majority of the state-of-the-art algorithms and approaches for semantic segmentation are still typically designed manually (He et al. 2021) and require profuse domain expertise to be constructed (Liu et al. 2010, Singaravel et al. 2020); which is expensive both in terms of computational and human resources (Siam et al. 2018, Cordts et al. 2015, Geiger et al. 2013).

A popular approach to tackle semantic segmentation problems is to use architectures of the so-called UNet type (Ronneberger et al. 2015a), which have a similar structure to an autoencoder but incorporate extra skip connections from the downsampling stage

to the upsampling one (Zhang et al. 2017a). UNet architectures are constructed manually following a specific set of rules that can be potentially automated through Neural Architecture Search (Ronneberger et al. 2015a) discussed later in this chapter.

Anomaly Detection and Imbalance: Another significant aspect of computer vision is anomaly detection, which deals with the "detection of deviation and divergence of anomalous samples from the normal ones" Minhas and Zelek (2019). It is concerned with recognising patterns of data (called anomalies), which appear to be out of the ordinary distribution (Chandola et al. 2009). Recognising these anomalies is crucial for various problems from a multitude of domains, including threat detection ² (Minhas and Zelek 2019).

Identifying anomalies can be especially hard in scenarios where the natural distribution of data involves high variance since most anomalies can be well concealed to look like natural outliers of the original distribution (Mery 2015). There exist anomaly detection algorithms that use supervised, semi-supervised and unsupervised learning, where the ability of unsupervised learning techniques is believed to be limited for analysing images (Minhas and Zelek 2019).

There are two major challenges when using supervised learning for anomaly detection: the *lack of labelled data* and *low anomaly instances* compared to the benign ones (Minhas and Zelek 2019). In the "data space", a popular way to deal with massive data imbalance, if possible, is to over-sampling the data in favour of the misrepresented class or classes using approaches such as SMOTE (Chawla et al. 2002), MWMOTE (Barua et al. 2012) or many others. As a result, new data points are generated, which follow the distribution of the presented dataset. With imagery data, oversampling is usually done using data augmentation (Taqi et al. 2018). More recently, generative AI such as MidJourney and DALLE provide even more creative and novel ways of upsampling data (Seneviratne et al. 2022, Bandi et al. 2023)

Another option is to undersample the benign examples, such that the distribution of anomalies is similar to that of benign ones (Anand et al. 2010). Researchers have recently addressed these questions with the idea of a critical dataset, which is the minimum number of samples needed for a model to capture the underlying distribution of data generation (Goodfellow et al. 2016b) using coresets discovery which is what is also employed in MIRA-ME (Chapter 5 (Har-Peled and Mazumdar 2004, Guo et al. 2022).

In the "model space", numerous techniques also exist to address limited data availability and data imbalance (Miao et al. 2019, Minhas and Zelek 2019). The model can be fine-tuned to the dataset at hand using transfer learning with an already trained model for a different dataset (Vercruyssen et al. 2017). A dataset often used for transfer learning is ImageNet (Appendix G) since it is one of the most extensive image datasets available (Deng et al. 2009, Torrey and Shavlik 2010). Transfer learning allows the model to begin

²<https://deeper-scan.com> (Accessed on 12.05.2020)

training at a potentially good starting point while training on the new data points to reach the global optimum more efficiently and effectively.

Another technique used to combat class imbalance is adjusting the biases of the output neurons for some of the classes (Givnan et al. 2022) or using weights in the loss functions to steer the model in giving underrepresented classes more attention (Zhu et al. 2018) as well as using loss functions like the Focal loss (Lin et al. 2017b), which are specifically designed to be as invariant as possible to class imbalance. For instance, adding weights to categorical cross-entropy loss for n classes can be done by changing the conventional categorical cross-entropy:

$$L = -\frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i)^T \cdot \log(\hat{\mathbf{y}}_i), \quad (2.2)$$

to include a weight term α for the n different classes:

$$L = -\frac{1}{n} \sum_{i=1}^n (\alpha \odot \mathbf{y}_i)^T \cdot \log(\hat{\mathbf{y}}_i), \quad (2.3)$$

where \mathbf{y}_i is a $c \times 1$ target vector containing a one for the c_i entry corresponding to the class of the i^{th} input image, and zero otherwise (that is, $\mathbf{y}_i \equiv \mathbb{I}_{c_i}$, where \mathbb{I}_{c_i} is the indicator function), $\hat{\mathbf{y}}_i$ is the output of the network (a $c \times 1$ vector of probabilities for the i^{th} input pattern to belong to each class), α is the $c \times 1$ vector of class weights correcting the class imbalance, and \odot is the Hadamard (element-wise) product.

Semi-supervised approaches for this problem usually attempt to estimate the underlying distribution and density function (Akçay et al. 2018). For example, Generative Adversarial Networks (GANs) minimise the difference between the images from the dataset and their generated reconstructed counterparts. This way, a larger distance from this learned data distribution at inference time "is indicative of an outlier from that distribution" (Akçay et al. 2018), which is an anomaly. This concept is utilised during the design of the unsupervised pipeline of the data selection approach shown in Chapter 5.

2.3 Multi-Objective Optimisation

Multi-objective optimisation refers to the process of optimisation of a system to solve a problem constituting two or more objectives, which is often the case with real-world problems (Deb 2001a)(Deb 2014). In the domain of computer vision and especially the one of defence, this is of immense importance since every single objective might be mission-critical and needs to be considered when an optimal solution is proposed and deployed (Lee et al. 2019).

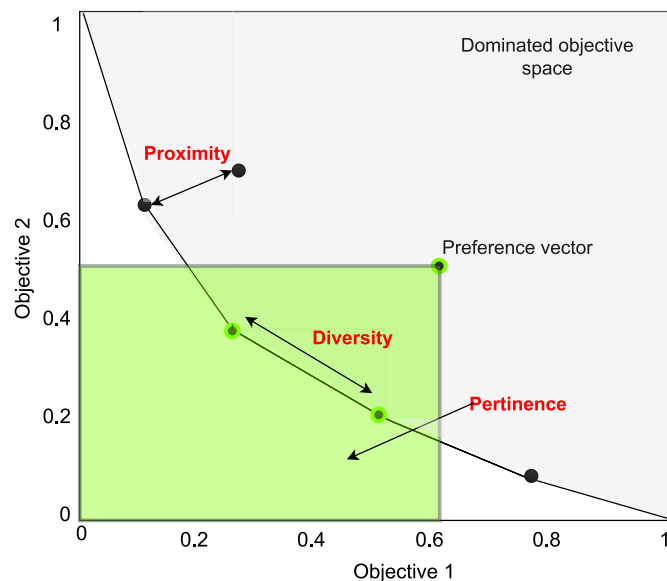
In single-objective problems, there is usually a single metric of the performance of a

solution (Deb and Tiwari 2008). This metric is often called the “fitness score” of a solution. In the case of problems with two or more objectives, a single-objective optimisation is inappropriate since a proxy of all would be required to represent the complexity of the problem.

On top of that, the objectives might be conflicting; thus, optimising for one may also mean compromising the other. This problem would require a solution that weights them in some way while ensuring the approach is not susceptible to getting stuck in local optima (Hwang and Masud 2012, Deb 2014).

Addressing this caveat requires the conversion of the solution to be an approximation set which consists of multiple candidate solutions with different combinations of objective values, instead of a single solution (Rostami 2014, Dimanov 2019). The quality of this approximation set is generally determined using the following three characteristics: proximity, diversity and pertinence (as displayed in Figure 2.3) (Rostami and Neri 2016).

Figure 2.3: The main characteristics of an approximation set, the dominated objective space and the Pareto front can be observed. The Pareto front in the image is represented by the line drawn between the solutions (black and green dots)



Pareto optimality: These properties are used to describe Pareto optimality, which is a term that describes the front (collection) of non-dominated solutions in a multi-objective space that depicts the trade-off of objectives (Censor 1977).

The concept of Pareto optimality originated in the domain of economics as a way to achieve a balance between maximisation of societal ophelimity and exhaustion (Luc 2008). It soon became applicable to many other use cases, including multi-objective optimisation (Ngatchou et al. 2005).

In multi-objective optimisation, Pareto optimality denotes the optimal trade-off of solutions considering multiple different (and often conflicting) objectives (Ngatchou et al.

2005). The solutions on the Pareto front are all non-dominated in the objective space.

Numerous summative measures have been used to describe the performance of a frontier of solutions compared to a known set of optimal solutions (the Pareto front) (Sandler and Smith 1982). These metrics measure the distance from the known Pareto front using metrics that reflect the three attributes of an attribution set (described at the beginning of sub-chapter 2.3).

A solution lies in the dominated space if at least one other solution fully dominates it. The other(dominant) solution has better values for all observed objectives. Thus, since the dominant one is better across all objectives, the dominated solution is usually discarded.

All non-dominated solutions form a front which is called the Pareto optimal front. It represents the most optimal approximation set for the given objectives (Censor 1977). Many algorithms and methods use an estimation of the Pareto optimal front to guide the process of optimisation, but this approach can often be deceitful since it is not usually possible to know what this front is beforehand. Thus, approximations can be inaccurate.

Characteristics of the Pareto front: Moreover, if the Pareto optimal front is known before the optimisation begins, the process may become superfluous since the process of optimisation is attempting to discover an entity, which may be already defined (Zapotecas-Martínez et al. 2018).

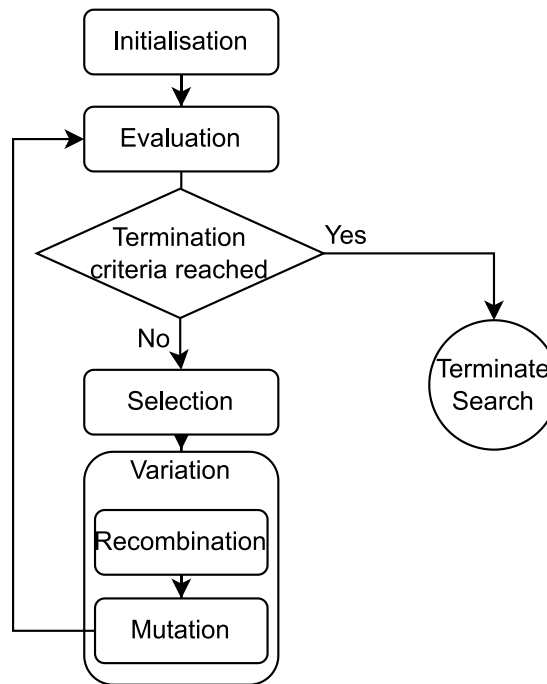
The first of the three characteristics in Figure 2.3 is called **proximity**. It serves as a proxy of how close the solution is to the Pareto front. The second one - **diversity** - is used to evaluate the distribution of solutions across the approximation set, which in Figure 2.3 illustrates as the distance between each solution (Woolley and Stanley 2014).

The last characteristic allows the decision-makers to specify a **preference vector**, which defines a region of interest. In this way, the decision-makers can directly impact the outcome of the process and influence it to look for solutions within the region of interest. This characteristic is called pertinence, and it is represented by the green shaded area in Figure 2.3 (Rostami and Neri 2016).

All three characteristics of an approximation set are designed to aid the decision-making process and make it easier for decision-makers to select a solution or group of solutions based on their requirements.

That is why multi-objective optimisation, in difference to single-objective optimisation, explores the optimal set of solutions and accounts for the intrinsic trade-off of different objectives, which in turn allows for a much more diverse and optimal set of solutions and provides a collection of solutions from which the decision-maker or client can choose a specific solution based on its objective values (Rostami and Shenfield 2012).

Figure 2.4: The process followed by most evolutionary algorithms with the major steps presented and an example of breaking steps into sub-steps is presented for **Variation**. Splitting variation into Crossover and Mutation is usually done for a subset of evolutionary algorithms called Genetic algorithms.



2.3.1 Evolutionary Computation

Next, let's turn to evolutionary computation, which is a collective term used to address a collection of biologically inspired optimisation algorithms, which often, to some extent, depict the process of evolution (Coello et al. 2007). More precisely, evolutionary computation draws inspiration from modern genetics and natural selection (Deb 2001b) (Dimanov 2019) in an attempt to discover a population of individual solutions (approximation set), which would be positioned as closely as possible to the global optima and ideally should lie on the Pareto front of optimal solutions (Ishibuchi et al. 2016).

The algorithms usually have five general steps to achieve this ambitious task, which can be subdivided into more detailed ones (Kachitvichyanukul 2012). The basic working principle of these algorithms is that they start with a population of random solutions. Then, through variation and selection, the solutions improve over time from generation to generation until a termination criterion is met (Kachitvichyanukul 2012).

Each step is detailed in the following sections, which cover the process more in-depth, as depicted in Figure.2.4.

Initialisation: In this step, a population of solutions is initialised, which usually involves constructing a random population of individuals, which have random values for each property from the search (often called "decision") space (Coello et al. 2007). The search space encompasses all possible values for all variables.

Evaluation: After the population of individuals has been initialised, it is evaluated to see how the different solutions perform for the given problem. During this step, the genome is transformed to its phenotype (Deb et al. 2002a), and then each individual (the set of decision variables) is taken and used as an input to some sort of objective function, which in case the problem is single-objective is often called the "fitness" function. After this is done, in most cases, a single value would be produced, which would be that individual's fitness score (objective score) for the problem (Liu et al. 2019b).

In the case of multi-objective problems, there will be more than one objective score, which can either be summarised into a single value or optimised simultaneously (Deb 2001b, Coello et al. 2007).

Termination: A termination criterion is predefined, which may be a maximum number of iterations (generations), the time elapsed, the maximum number of evaluations (objective functions executed) or simply a desired objective score (Safe et al. 2004).

Selection: Then, the selection step takes the evaluated individuals and chooses which ones should continue and which should be discarded.

In this step, prevalent approaches include tournament selection (Blickle and Thiele 1996), elitism (Yang 2007), ranked selection (Blickle and Thiele 1996) and many more.

This step draws inspiration from the process of natural selection and survival of the fittest in nature (Fogel 2000). Choosing the proper selection can enormously impact the overall algorithm's performance (Nayyar et al. 2018). This step is heavily reliant on the chosen objective or objectives, and it also depends on the mechanism to treat these objectives. There are two stages of selection since the parents that continue the cycle are picked, but sometimes survivor selection also needs to be conducted (Nayyar et al. 2018).

Variation: Although selection is an essential step of the evolutionary process, arguably the quintessential one for evolutionary computation is variation. The variation process differs from one family of evolutionary algorithms to another, but there are two main sub-operators of achieving variation.

The first one is called **crossover**. It is often seen in genetic algorithms since it aims to simulate biological reproduction. It features the combination of genomes (chosen candidates from the selection) in various ways, including one, two or k-point crossover, which means that the genome (genotype representation of the candidate) is split into one, two or k-points.

Then, the offspring (new individuals produced as part of the updated population) are created by recombining these pieces from different parent candidates. There are different types of crossover. K-point crossover is just one way of achieving recombination. It can also happen through uniform crossover (performed to sample genes and chromosomes from both genomes uniformly), Edge Recombination Crossover (Deb et al. 2007), Differential evolution crossover (Pampara et al. 2006) and many more.

In general, there exist two main approaches. These are:

1. Using just mutation, which means introducing random changes to the individual's genome (the set of all decision variables for the particular individual) (De Falco et al. 2002).
2. Using mutation in combination with crossover (or recombination), which is usually done in genetic algorithms, which aim to depict how genetics work more closely (Stanley et al. 2019a).

2.3.2 Evolutionary Multi-Objective Optimisation

Evolutionary algorithms, by design, use a scalar value which determines the fitness score of an individual solution (Barbiero et al. 2019). To facilitate multi-objective optimisation in the context of evolutionary algorithms, the problem is usually stated as *maximise* $F(x) = (f_1(x), \dots, f_n(x))^T$ where x is the vector of decision variables making up a proposed solution (Zhang and Li 2007).

So, a wrapped objective function is constructed, which uses the different scores for all objectives and aggregates them into a single number. Since usually there is no point in the decision space which maximises all objectives simultaneously, the best solutions can be defined by their Pareto optimality (Zhang and Li 2007).

The approach of Zhang and Li (2007) uses decomposition and optimises each objective separately by defining each problem as a subproblem and then using information about neighbouring subproblems during their separate optimisation. The crucial part of this approach is that the population consists of historically the best candidates x for each distinct objective, whereas the non-dominated solutions are stored separately (in an archive-like structure), termed external population (EP).

On the other hand, NSGA-II (Deb et al. 2002a) filters similarly ranked solutions based on crowding distance. The crowding distance for each non-dominated solution is calculated based on predefined objective criteria reference points, similar to computing the cHV.

2.4 Automated Machine Learning

As pointed out by Yao et al. (2018), the pivotal book for machine learning of Mitchell et al. (1997) starts with: "Ever since computers were invented, we have wondered whether they might be made to learn. If we could understand how to program them to learn - to improve automatically with experience - the impact would be dramatic".

Automated machine learning (**AutoML**) is a field of machine learning which aims to decide on machine learning models "in a data-driven, objective, and an automated way" (Hutter et al. 2019a).

With the increase in available computation (Moore et al. 1965), researchers started exploring the idea of automating the optimisation process of CNNs. Hence, the latest advancements in the state-of-the-art in computer vision are a product of automated machine learning (AutoML) (Real et al. 2017, Tan and Le 2019a, Tanaka et al. 2020).

In the domain of neural computation, AutoML is a process of discovering the correct hyperparameters (e.g. layers, number of neurons per layer, order of layers, types of layers) and parameters (weights and biases) of a neural network (Hutter et al. 2019b).

2.4.1 Neural architecture Search

Even though the use of machine learning has grown drastically in the last decade, the need for experts and computational resources to design, evaluate and train these machine learning models has also risen dramatically, which "hinders the development of deep learning in both industry and academia" (He et al. 2021).

The rise in popularity of AutoML is a natural response to address the high demand for expertise and computational resources for deep learning in industry and academia (He et al. 2021, Stanley et al. 2019a). The automation involves not only the NN training but also the design of architectures, which is known as **Neural Architecture Search** (NAS) (Stanley et al. 2019a, Elsken et al. 2018b). Many different types of algorithms are used for neural architecture search, but the two most popular ones are reinforcement learning and neuroevolution (Stanley et al. 2019a).

Multiple different algorithms and approaches have been proposed to do just that. The first significant one is the work of Zoph and Le (2016). Soon after, various reinforcement learning (Qin and Wang 2019, Tan and Le 2019b), Neuroevolution ones (Lu et al. 2019, Miikkulainen et al. 2019, Real et al. 2019, Stanley et al. 2019a) and other approaches started to emerge.

While most NAS approaches treat neural network optimisation as a black-box process, DARTS uses gradient descent optimisation over a continuous search space of architectures. They aim to find complex building blocks with intricate graph topologies that are not restricted to any family of blocks (Liu et al. 2018). To do so, the authors represent the connectivity of the different layers using a directed acyclic graph, which has inspired the encoding design in Chapter 4.

One of the more significant breakthroughs in the field was in 2017 when Real et al. (2017) reached state-of-the-art performance on a widespread computer vision problem and outperformed domain expert-designed models. To do this, the authors used a simplified graph to represent architectures during encoding, meaning mutation could be used on each node separately and could happen more than once. However, with this breakthrough, some standard NAS caveats became even more apparent. Mainly the computational hunger of these algorithms.

Finding an architecture is an essential step in both research and development when a researcher or practitioner attempts to tackle a novel problem (Perez et al. 2019). A severe limitation for both industry and research is the 3150 GPU days requirement of AmoebaNet (Real et al. 2017 2019) and the 9000 GPU days one of NASNet (Qin and Wang 2019). Furthermore, these massively computational hungry algorithms are aimed at performing NAS on a relatively low-dimensional dataset known as CIFAR-10 (discussed and used in Chapter 3).

Moreover, the main computational power of the algorithm is not even used for searching for the architecture, but rather it is soaked up by the expensive evaluation process, which includes training the architecture to a good standard before being able to score it (Real et al. 2018).

This problem has spawned an entirely new NAS sub-field dedicated to developing proxy scores that can potentially allow such algorithms to run exponentially faster (Abdelfattah et al. 2021) and evaluate an architecture by requiring only one batch evaluation (Abdelfattah et al. 2021, Mellor et al. 2021) or in some cases, the proxies can be data agnostic and not need the architecture to see the data it will be trained with (Tanaka et al. 2020). These scores are discussed in more detail in Section 2.4.3.

2.4.2 Neuroevolution

Neuroevolution is a type of AutoML approach (dating back to the 20th century), which relies on evolutionary optimisation to optimise neural networks (Lehman and Miikkulainen 2013). Neuroevolution algorithms are a specific type of evolutionary algorithms which aim to optimise neural networks in some way (Stanley et al. 2019a). Some Neuroevolution approaches focus more on neural architecture search (finding an optimal neural network architecture) (Real et al. 2017) while others attempt to optimise the weights (Jenkins 2006, Lehman and Miikkulainen 2013) or hyperparameters of the network (Real et al. 2018).

At first, researchers were looking at optimising weights and other parameters of neural networks. However, with the increase of available computational resources, research started focusing on looking for hyperparameters and neural architectures (Floreano et al. 2008). Soon Stanley et al. (2009) came out. It allowed researchers to consider even bigger neural networks and to explore the use of neuroevolution in computer vision as well (Real et al. 2017). In 2018 Real et al. (2017) was published. It showed the potential of neuroevolution to optimise large CNNs. Even though it used the evolutionary approach to optimise a vast number of hyperparameters, the parameters themselves (the weights and biases) were still using a conventional gradient-based approach, which made AmoebaNet (Real et al. 2018) unfeasible to train in a normal research setting, since for grayscaled CIFAR-10 it took 3150 GPU days to discover the best architecture. Some other examples

have attempted to apply neuroevolution in the context of computer vision, like Genetic CNN (Xie and Yuille 2017). To the best of the author's knowledge, very few, if any, have attempted to optimise both hyperparameters and parameters only through the means of neuroevolution before fine-tuning.

A significant breakthrough in the field occurred in 2017, when Real et al. (2019)'s **AmoebaNET** reached state-of-the-art performance, outperforming architectures designed by experts. The AmoebaNet is a Neuroevolution algorithm (Real et al. 2019). In such approaches, decoding the NN (the individual in an evolutionary algorithm context) is added as an extra step before the evaluation, whilst encoding the individual back to genotype is added after the evaluation (Stanley et al. 2019a). For its encoding, the authors use an encoding similar to the one of DARTS (Liu et al. 2018).

The process of neuroevolution only differs from the usual evolutionary approach discussed in 2.3.1 by the addition of encoding and decoding as depicted in Figure 2.5. In reality, the initialisation, variation and evaluation also have differences (Stanley et al. 2019a). They serve the same purpose as the steps in the standard evolutionary algorithms, which makes possible the use of a neuroevolution algorithm using a library or set of tools designed for typical evolutionary algorithms.

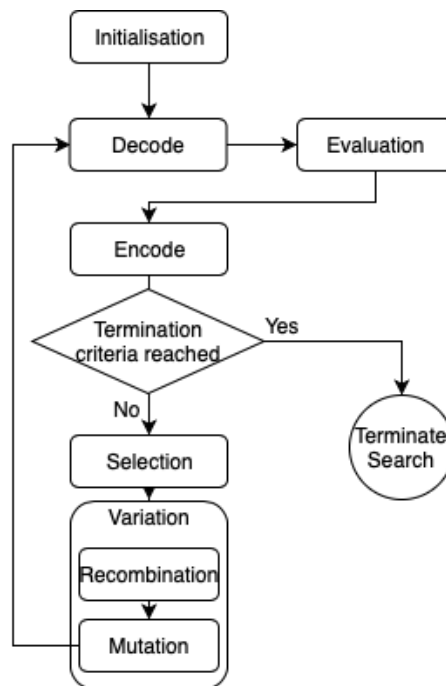
Recently, indirect encodings (Hadjiivanov and Blair 2016) (closely inspired by how DNA works (Miikkulainen et al. 2019)) have shown promising results. Neuroevolution has been successfully used for autoencoder optimisation in a range of studies (Charte et al. 2020, Sereno 2018, Alvernaz and Togelius 2017); however, a limitation of these approaches is that they typically focus on evolving a NN with dense activation layers (e.g., in Charte et al. (2020)'s EvoAAA) instead of convolutional and pooling layers, which have shown great potential for computer vision (Gu et al. 2018).

Next, some key concepts for neuroevolution are discussed. The subsection concludes with a brief explanation of how everything ties together.

Neuroevolution, similar to other evolutionary algorithms, needs a **search space** to be defined, which is the decision space which the algorithm will transverse in an attempt to find the best decision variables (Elsken et al. 2018b). This search space is defined by the set of possible values for each decision variable, and it can grow to exorbitant size (Elsken et al. 2018b). If architectures need to be discovered, some constraints have to be put to limit the search space because of its vast dimensions. Also, the available computational resources to compile the genome of the actual neural network (Liu et al. 2018). Usually, this is done through the incorporation of a maximum number of layers, a predefined number of cells (which have a predefined maximum number of layers inside them (Tan et al. 2019, Real et al. 2018)) or other measures.

The **initialisation step** in neuroevolution is another possible caveat of the approach. As discussed in Pretorius et al. (2018), Sun et al. (2019) neural network initialisation can greatly impact the performance of the neural network even after it converges (Goodfellow

Figure 2.5: A high-level view of the neuroevolution process.



et al. 2016b) . What is more, several strategies are common when attempting initialisation for neural architecture search, which include starting with fundamental structures and then building to more complex ones (Sun et al. 2019) or splitting the process of optimisation of the architecture into two stages (which is one of the most common approaches Stanley et al. (2019a)) and having micro and macro searches.

Micro search denotes the process of looking for the best neural architecture cell, which includes looking for the best types of layers, how many nodes are in a layer, connections between layers and much more (Lu et al. 2019) . **Macro search**, on the other hand, attempts to optimise the whole network while searching for the right layers and connections simultaneously (Zoph and Le 2016). (Lu et al. 2019).

There exists a hybrid search (sometimes referred to as block search and other times just labelled as macro search also) which uses predefined blocks, usually discovered by micro search and arranged them in order to (Real et al. 2017) find the best optimal architecture based on stacked cells (finite in this case) (Lu et al. 2019).

The motivation behind this approach is based on most state-of-the-art human-designed approaches which use a similar methodology. For example, looking at Resnet (He et al. 2015) , InceptionNet (Szegedy et al. 2015b) , VGG (Simonyan and Zisserman 2014) and other popular architectures and their families of algorithms, what can be seen is that there is some cell (as explored in more detail in 2.2), and then it is just stacked x amount of times. For example, ResNet 18 has 18 ResNet blocks/cells, and ResNet 34 is the same, except it has 34 ResNet blocks/cells.

In the case of using macro and micro architecture searches, there are two separate

pipelines that require separate initialisations.

Encoding and decoding are required to use suitable genetic representations during variation, initialisation and selection (Floreano et al. 2008). These representations should allow the genetics operations to easily change single or multiple decision variables without breaking the phenotype.

There are multiple different approaches to encoding and decoding, and in general, the decision variables (that the genome is composed of) can be directly (Floreano et al. 2008) or indirectly (Hadjiivanov and Blair 2016) encoded. Recently, indirect encodings have shown promising results, and they are modelled after our understanding of how DNA works (Miikkulainen et al. 2019).

In **direct encoding**, the neural network parameters/hyperparameters are represented by a single decision variable in a one-to-one mapping fashion (Floreano et al. 2008). For example, if neuroevolution is only used for optimising weights and biases of a fixed architecture, then the concatenation of all parameters would represent the genome (Floreano et al. 2008).

A popular approach for direct encoding is to use bit-encoding, which means encoding all variables as a sequence of bits and then variation through bit-flipping (Back et al. 1997) to conduct the optimisation. Several approaches have attempted adaptive encoding motivated by the vast search spaces available and the prerequisite of knowing all possible decision variable values before the process begins.

The work of Schraudolph and Belew (1992) suggests using binary encoding to encode the crucial bits of the weights and, after they converge to reuse the same bits for correcting and encoding the less critical parameters, thus implicitly staging the training process. Motivated by this idea, RAMOSS (Chapter 4) aims to make encoded connections compatible with bit operations.

Evidence of the success of direct encoding is the NEAT algorithm (Stanley and Miikkulainen 2002a), which has been employed for classification tasks, reinforcement learning problems, game development and many other scenarios.

While direct encoding has been widely utilised, more complex representations were needed to depict complex architectures since, in literature, it is argued that direct encoding has some drawbacks when applied for neuroevolution in larger networks (Floreano et al. 2008).

At first, **indirect encoding** Kitano (1990) showed superior performance over competitive direct representations, but it was later discovered that the causality of the different performance had much more to do with the initialisation than the representations themselves (Siddiqi and Lucas 1998).

Some of the other approaches in indirect encoding include cellular encoding (Gruau 1994a), adaptive growth through differential equations (Husbands et al. 1994) and many others. The recent advancements in the field have been focused on representing genetic

processes and constructing digital DNA of the ANN through the encoding, which has been attempted using gene regulatory networks (Kumar and Bentley 2003), analogue genetic encoding (Mattiussi et al. 2008) and a popular one called connective compositional pattern-producing networks (Stanley et al. 2009).

Evaluation: After the decoding is complete and the neural network is constructed, follows the evaluation phase. Evaluation not only in the case of neuroevolution but also for any evolutionary computation relies heavily on defining the objective function or functions. This objective function or functions (in the case of multi-objective problems) are used to determine the performance of any individual(solution).

They map an individual's representation in the decision space to its position in the objective space. There are different approaches as to how evaluations should be done in a neuroevolution algorithm (Stanley et al. 2019a). One approach is to run the whole network together with the weights as parameters of the evolutionary algorithm. This way, the algorithm handles the training, neural architecture search and hyperparameter optimisation (Miikkulainen et al. 2019).

However, by far the most popular approach is to train the constructed network for e amount of epochs on the training set (Real et al. 2018, Lu et al. 2019) and then evaluate it using the evaluation set.

This process produces a metric or metrics that can serve as objective scores, determining how good the particular genome is (Lu et al. 2019). Some works train for some predefined number of epochs (Davison 2017), others that spend a particular time on the training or, in rare cases, wait until the architecture converges (Real et al. 2019, Zoph and Le 2016).

Here, all hyperparameters of the network come into play during the training. Usually, the loss or the accuracy serves as the objective or one of the objectives of the algorithm (Aly et al. 2019), which is also **the most expensive** (in terms of computational resources) step during the whole process (Real et al. 2019) and hence, there exist multiple different methods to speed it up (Elsken et al. 2018b, Mellor et al. 2021).

These methods range from using deep learning tricks like pruning the architecture (Siebel et al. 2009) to reduce the number of parameters, using early stopping (Assunção et al. 2019) to avoid spending too much time on unpromising architecture (Aly et al. 2019) or, in some cases, using a proxy score (discussed below) to determine how good it is (Mellor et al. 2021).

After all individuals are evaluated based on their objective values, the **selection** phase takes place. The purpose of selection is to exploit the search space. During this stage, a portion of the evaluated individuals is selected to continue to the next generation (Stanley et al. 2019b, Goldberg and Deb 1991).

Following the selection of the successors, **variation** is applied. The purpose of this step is to explore the search space, and it does so by using mainly two operations (Fogel

2000) , namely: crossover (Jansen et al. 2002) and mutation (Koenig 2002). Variation in neuroevolution is slightly more complex than the one in normal evolutionary algorithms. In this case, the chromosome is an encoded neural network, and there are certain constraints to consider when doing crossover and mutation Stanley et al. (2019a).

However, neuroevolution also has several profound **drawbacks**, which make it either unfeasible or simply impractical for specific problems (Eaton 2015). It does not scale well to high-dimensional representations (such as raw images) (Alvernaz and Togelius 2017).

Evolutionary algorithms by nature are computationally intensive even for small and relatively simple problems (Zhou et al. 2020) . They usually utilise the parallelism of tasks, which makes them substantially faster (Miikkulainen et al. 2019) . One problem with neuroevolution, in particular, is that because the evaluation phase resorts to training neural networks, which also rely on large parallelisation of tasks, the evaluation of different individuals needs to happen in an element-wise fashion. The algorithms exploit multiple powerful GPUs and distribute the training of the models to them to make it feasible while making neuroevolution extremely time and computationally-consuming (Real et al. 2018).

Most NAS approaches focus on image classification - for instance, the AmoebaNet family (Real et al. 2017 2018), NASNET (Zoph et al. 2018), LEMONADE (Elsken et al. 2018a) and many others (Real et al. 2017, Lu et al. 2019, Galván and Mooney 2021)-but only a small percentage of them were indeed used for real-world applications (Galván and Mooney 2021). This is mainly due to the enormous computational cost of the training process in these approaches, even for relatively small datasets (Stanley et al. 2019b).

To overcome this drawback, some approaches split the process into two sub-problems. First, they focus on finding a repeatable cell, like in the ResNet (He et al. 2016a) or in the InceptionNet blocks (Szegedy et al. 2015c) (a process termed *micro search*); and then on optimising how to stack these cells (*macro search*, (Lu et al. 2019, Qin and Wang 2019)). Another popular approach is to use a *SuperNet* (Liu et al. 2018) containing all possible optimisation operations; then, candidates are constructed as sub-graphs of this net (Chu et al. 2021). This approach often implies weight-sharing; hence, network candidates benefit from training their peers (Chu et al. 2021).

Regardless of the search strategy utilised, state-of-the-art methods for iterating over the search space mainly consist of evolutionary algorithms (Galván and Mooney 2021), reinforcement learning (Tan and Le 2019a), Bayesian optimisation (Shahriari et al. 2015) and other gradient-based methods (He et al. 2021). Perhaps the two most popular approaches are neuroevolution and reinforcement learning (Stanley et al. 2019b). More broadly, the application of evolutionary algorithms for AutoML, particularly for NAS, has yielded encouraging results (Real et al. 2017 2018, Elsken et al. 2018a). These algorithms are often considered one of the most promising directions in terms of efficiency for this type of problem (Radiuk and Kutucu 2020, Stanley et al. 2019b).

Notably, the most expensive step in current AutoML approaches, regardless of their type, is the evaluation of the candidate nets since they have to be effectively trained to generate realistic estimates (He et al. 2021). Avoiding the training step drawbacks has been recently battled with the weight-sharing, and SuperNet approaches as mentioned above (Liu et al. 2018). However, recently, some works suggested using pruning scores for networks that can be used as proxies for the trained performance of a network (Abdelfattah et al. 2021, Mellor et al. 2021, Guo et al. 2019b). However, most of these proxies are still only compatible with classification tasks and are only reliable to a certain extent (Mellor et al. 2021).

In the case of one of the most popular algorithms, it took 3150 GPU days to discover a network for a standard benchmark dataset (Real et al. 2017) and unfortunately, due to tight deadlines and demand for iterative improvements in industry, it has limited applications.

2.4.3 Proxy scores

Solving any of the presented problems so far requires careful consideration for choosing the right performance metric for the problem at hand. To do so, the chosen measure must correctly assess how well the defined task is solved and account for the data distribution, noise and other data-specific challenges.

In addition to the plethora of classic metrics it is worth stressing the emergence of new metrics which are directly relevant for this thesis' contributions, discussed in the following chapters (Wu et al. 2021, Mellor et al. 2021).

A breakthrough in this niche area was recently presented in the paper "Neural Architecture Search Without Training" (Mellor et al. 2021), which explores the idea of using such custom **performance proxy metrics** to score neural architectures from NAS search space without even training them. According to numerous studies (Abdelfattah et al. 2021, Wu et al. 2021) the metric has a positive correlation when tested for similarity with the performance of a trained network.

Studies leveraged the fact that the performance of a network can be estimated upfront by employing it in the loop of neural architecture search. The most (computationally) step in the whole neural architecture search process is the training of the networks until some type of convergence. A proxy metric that can bypass this step and assign an objective score to the given network without training can revolutionise the whole process by potentially speeding up the process exponentially exponentially (Abdelfattah et al. 2021).

Such approach is **NASWOT** (Mellor et al. 2021) approach that uses a single batch of data to determine how good an architecture is for a given dataset. The authors specify that their approach best (if not only) works with rectified linear activations; using the single batch, the approach consists of processing images through the network and identify linear

binary activation regions.

Next, these binary regions are compared based on the Hamming distance between them to determine how similar the activations for different inputs are. Their underlying intuition of the approach is that the more different the activations for different inputs are, the better the network is.

This difference score can then be fed to any NAS method, giving them a tremendous speed advantage over conventional approaches. Although results might not reflect the true performance precisely since the correlation is not above 90%, they still produce a good estimation of the model's performance. It can also be mitigated by using approaches such as neuroevolution, where often the result is a population of solutions rather than a single candidate network.

SYNFLOW: In difference to NASWOT and other proxies, Synflow (Tanaka et al. 2020) relies on the synaptic flow of different neurons to estimate how important a specific neuron is for the propagation of the signal in the network. Originally, authors employed this score to prune the network by removing neurons with low scores for their inward and outward signals. The calculation of the score takes seven straightforward steps.

The unique thing about this approach is that it is data agnostic, in the sense that to calculate it no data is needed, but rather just the spatial dimensions (shape) of the batch suffice. The first step is to map all weights through a modulus operator to convert them to absolute values. Then, a vector of ones of the same shape as the expected input is provided as an input. Next, all outputs of the selected layers are extracted and summed up to result in a single number, which is the pseudo-loss function (step 4).

The fifth step consists of computing all gradients of the selected layers with respect to this pseudo-loss for the different outputs. The final step (step 6) of the process is to multiply the weights by the backpropagated signals. Lastly, the weights are reverted to their original signs using the signs stored when the modulus operation from step one is conducted.

In this way, the network is pruned dynamically decreasing the magnitude of unimportant weights and disregarding them, but avoiding layer collapse (Tanaka et al. 2020). To use this pruning for NAS, some works sum up the average of the scores found from step 6 of the process and come up with a single number which can be used as a proxy for the capacity of the model (Abdelfattah et al. 2021).

Having such approaches has long been one of the main problems in the field of NAS and finally, having such proxies with a high correlation to performance is groundbreaking and could allow researchers and practitioners to use many old and new algorithms more efficiently and to produce results magnitudes of time faster.

2.5 Applications to Concealed Threat Detection

The need for concealed weapon detection systems is rising, and security is currently a significant global concern (Mahajan and Padha 2018). Researchers are attempting to use various methods to identify, classify and detect threats in digital and X-ray images; often, these threats are concealed, and detecting them is significantly more challenging than other computer vision problems (Grega et al. 2016).

There are numerous ways to detect concealed weapons, such as millimetre wave technology (Goenka and Sitara 2022, Li and Wu 2022), radar (Zhang 2022) and, most often, X-ray machines (Miao et al. 2019, Akcay et al. 2018, Nguyen et al. 2022). The X-ray machines used for this purpose are usually dual-channel with high and low-intensity X-rays, which enables them to penetrate through different types of materials (Sidky et al. 2011). This allows images to be colour-coded based on their densities, which facilitates visual inspections (Abidi et al. 2006). Recently, computer vision algorithms have been tasked with automating the vast majority of the imaging data coming from these various media sources, aiding human operators by automatically flagging potential threats or enhancing the images to make a threat more visible (Miao et al. 2019, Akcay et al. 2018, Rostami 2014).

This work focuses on the medium coming from X-Ray or CT scans. Usually, these techniques are used in tandem with an image processing pipeline, which analyses the scans and determines if a threat is present, sometimes what the threat is, and even where it is (Miao et al. 2019). These generated images are significantly different from the conventional RGB images generated from CCTV or other digital cameras (Mery and Arteta 2017).

Multiple challenges must be considered to detect threats in such images, including, but not limited to, the following. First, incorporation of information the colour hold (usually colours in X-rays and CT either signifying density (Hayler et al. 2019) or an object, the object material (Aichert et al. 2012) or some different physical property (Mery 2015)). Second, objects in such scans are densely packed together, which makes it harder to establish where one object ends, and another begins (Xia et al. 2021).

Moreover, there may be (and often are) overlapping objects, which appear to occupy the same pixels owing to the way the X-ray waves penetrate through objects (Röntgen 1895). All these factors and many more contribute to the noisy and increasingly complex background of such images and the data itself (Miao et al. 2019). Digital image processing to identify such threats in the complex background has been considered in existing literature (Hussein et al. 2016), and systems that assist operators are often considered (Tiwari and Verma 2015), because their results are usually not reliable enough to allow for a fully automated solution.

Some methods focus on these properties and attempt to use novel attention mech-

anisms designed to negate their adverse effects on model performance, such as the hierarchical refinement (Miao et al. 2019) and the Selective Dense Attention Network (Wang et al. 2021). Alternatively, additional pre-processing steps aim to aid models in “seeing” concealed objects by separating the different layers of the image and analysing the features separately before piecing together a full prediction (Mery et al. 2015).

Standard convolutional neural networks (CNNs) are typically used in these scenarios, often by simply naively applying already well-performing architectures with other types of visual images distinct from X-ray ones (Akçay et al. 2018, Hassan et al. 2020). However, some studies also leverage conditional generative adversarial network (GAN) to optimise the generator and the discriminator in the network simultaneously. For example, (Akçay et al. 2018) uses an encoder-decoder-encoder pipeline based on Deep Convolutional Generative Adversarial Network (DCGAN) (Radford et al. 2015) to recognise anomalies in X-ray images. Another popular approach is using transfer learning and retraining an already well-performing architecture on the new X-ray domain (Hassan et al. 2020).

Overall, a caveat of these approaches is that insights and techniques used with visual datasets may be suboptimal when used with other media types, such as X-rays (Mery 2015). An additional difficulty is the inherent imbalance present in the distribution of X-ray data for anomaly detection (Dumagpi and Jeong 2020). Most datasets contain a plethora of benign samples, and only a handful of anomalous ones (Miao et al. 2019, Mery 2015, Akçay et al. 2018), posing challenges for models in optimal generalisation while avoiding biased predictions in favour of the benign classes (Miao et al. 2019).

Creating entirely new models and techniques specifically for the x-ray domain is highly time-consuming. Moreover, detaching X-ray vision research from the advancements in the visual domain would hinder the joint progress of both fields. Thus, a potential workaround is to use transferable automated machine learning (AutoML) methods discussed earlier in the chapter (Section 2.4) (Xue et al. 2019).

In particular, neural architecture search (NAS) approaches are neither bound to the specific data at hand, nor to architectural paradigms used in visual datasets. Unfortunately, many state-of-the-art NAS algorithms are only viable for small dimensional datasets (such as MNIST and CIFAR 10), since they require a tremendous amount of computational resources. Moreover, the data loading and processing techniques are intertwined with the approach in some datasets, so they cannot effortlessly scale-up to high-dimensional X-ray data in a “plug-and-play” fashion (He et al. 2021, Real et al. 2017, Lu et al. 2018, Wang et al. 2021). As the interest in the field has grown, more and more research has attempted to fill this research gap and provide alternative solutions to deal with the problem, such as Dimanov et al. (2021), Zhou et al. (2020), Assunção et al. (2019).

Radio frequencies (Nikolova and McCombe 2015), radars (Kim et al. 2018) and electromagnetic radiation detectors (Bassen et al. 2019) are also widely used for concealed

weapon detection, but the problem with these approaches is that they are susceptible to noise and interference. They also often require active use by an agent on people one by one, which may hinder the traffic of people through secure locations and cause congestion (Mery 2015). The use of handheld devices with radar and ultrasonic detection has also been explored, but a limitation of this approach is that such devices could not work for long-distance detection (Nacci and Mockensturm 2001).

According to Grega et al. (2016) the problem can be solved by deploying smart cameras, which can identify an object in any given frame, then track that object and detect any form of concealed weapon. Thus, the threat detection problem (concealed or not) has become a computer vision problem.

The current limitations in the field are present mostly because threat detection in moving objects and under various lighting conditions is still inaccurate and susceptible to deception by different outside factors, such as types of clothing, lighting or continuous movement (Parande and Soma 2015, Olmos et al. 2018, Hussin et al. 2012).

Researchers have tried to alleviate and tackle some of these problems by using automated or semi-automated systems, which, with the help of artificial intelligence in the form of various machine learning approaches (Mery 2015), attempt to detect these threats from different types of signals (Rostami 2014) or imagery data (Hayler et al. 2019, Miao et al. 2019) and can promptly inform an operator of the given threat.

The need to process these immense data and the fact that many machine-learning computer vision tasks have recently been delegated to neural computation and, more specifically, the use of CNNs (Krizhevsky et al. 2012) has led to the idea of applying these approaches to the problem of concealed weapon detection. Active research is already underway to detect different threats, such as knives and guns (Olmos et al. 2018, Gelana and Yadav 2019), bombs (Majeed et al. 2018) or others (Dutta et al. 2018) with the use of CNNs. Based on the results, this research field shows tremendous prospects.

There are multiple different scenarios, where such computer vision algorithms may be employed, which range from facial detection of people carrying threats (Kamble et al. 2020), behavioural analysis of people and body language (Yang et al. 2021) to automatically discovering threats in X-ray and CT imaging (Miao et al. 2019). Some of these scenarios deal with conventional imagery data, so transfer learning (Chouhan et al. 2020) approaches are being employed to aid the overall performance of security systems, while others (such as X-ray and CT scanning) rely on substantially different processes to extract and analyse features most efficiently.

X-ray imaging enables the visualisation of the insides of objects (Mery 2015), which allows operators to explore the contents of a scanned item in a non-intrusive manner. Thus, this technology is largely used in airports (Liang et al. 2019b), crowded areas (Bhargava et al. 2007) and other high-security places.

One of the most popular methods for X-ray imaging, as already mentioned, uses

Figure 2.6: SIXray sample with multiple overlapping threats



dual-wave scans (Feder et al. 1977, Cherepennikov et al. 2015). In essence, three separate images are produced for each scan by shining a high-intensity and a low-intensity wave (Martin and Koch 2006). The waves complement each other when producing an X-ray scan, because they penetrate through different materials, thus producing a high-resolution image using the whole composition of the scans (Wetter 2013).

Atomic number analyses complement the two X-ray views, and an entity known as a z-map is generated, presenting a feature map containing the atomic value number per pixel of the images (Mery 2015, Gil et al. 2011). Because the images contain three channels (high and low intensity and the z-map), they can be integrated with conventional models designed for digital RGB images (Abidi et al. 2005).

These values are usually then represented as RGB values when preprocessing the input images, which eases human interpretation. Much of the geometrical and affine transformations for optical, as well as X-ray imagery work in the same way; thus, numerous transferable skills of computer vision experts work with optical data (Mery 2015).

Medium-specific processes and challenges exist especially in X-rays, including, but not limited to, the ones discussed above. One such obscure challenge is the attenuation of X-rays when passing through objects (Olmos et al. 2018). This is observed when X-ray imaging provides an image that contains information on object thickness and density (Mery 2015).

However, the choice of topology and other hyperparameters requires expertise in multiple different domains, so this neural network architecture is difficult to discover. Nevertheless, the increased research interest in using AutoML for computer vision problems has already delivered state-of-the-art results for multiple problems (Stanley et al. 2019a).

Thus, to conclude this chapter, the use of Neuroevolution, reinforcement learning and similar approaches to discover the correct CNN architectures seems promising and insightful for dealing with problems in the context of threat detection (Real et al. 2018). Moreover, researchers from Durham university Akcay et al. (2018) are already exploring the use of highly complex GANs that compete with each other to detect anomalies, even in unseen data.

This interest has led to an influx of novel X-ray datasets featuring concealed threats in recent years (Mery et al. 2015, Miao et al. 2019, Isaac-Medina et al. 2021). These datasets are designed to foster the upcoming automation of security systems, which would allow for security personnel to prevent potential disasters more efficiently and effectively (Mery 2015). A particularly relevant dataset is SIXray (Miao et al. 2019), which contains multiple distinctive threats that can appear simultaneously and exhibit unique properties, representing the domain's real-world data. Some of these properties stem from the penetrative nature of X-rays, such that some objects might be obscured by other benign ones, or that multiple objects of interest can be stacked on top of each other (see example in Figure 2.6) (Wang et al. 2021).

Chapter 3

MONCAE - Multi-Objective Neuroevolution for Convolutional Autoencoders

Before proceeding with the applications of AutoML to threat detection in Chapter 5, this chapter explores how to foster the efficiency of AutoML algorithms for computer vision.

One of the main drivers of this field has been the tremendous amount of available data, which has been stored and is accumulated daily worldwide (Oussous et al. 2018). This paradigm, known as Big Data (Lee 2017), has enabled multiple areas (including machine learning) to thrive by using this valuable resource. Chapter 2 suggested that existing AutoML studies in computer vision mainly focus on discovering image classifiers rather than other niche architectures, which seems to point to a major research gap in AutoML since recent techniques such as vision transformers, stable diffusion, semantic segmentation UNETs and even text-to-speech algorithms such as Whisper rely on autoencoder-like structures. Hence, this chapter addresses the research gap by presenting a new Multi-Objective Neuroevolution method for optimising Convolutional Autoencoders (MONCAE).

Remarkably, this is the first attempt, to the best of the authors' knowledge, to conduct a neural architecture search in the context of convolutional autoencoders as well as the first study to use a hypervolume indicator in the context of neural architecture search for autoencoders.

Results show that images were compressed in the bottleneck layer of the autoencoder by a factor of $\times 10+$ while still retaining enough information to achieve satisfactory image reconstruction and classification for the majority of the tasks. Thus, this new approach can be used to speed up the AutoML pipeline for image compression. It can also be integrated to discover autoencoders for newly surfaced stable diffusion, transformers or even generative adversarial networks.

This chapter focuses on one of the most promising approaches for automated machine learning - neuroevolution. Neuroevolution is chosen since it allows for a set of solutions to be generated which can be used to explore the trade-off between several different objectives (Kelly et al. 2023, Lu et al. 2019). One of the downsides of this approach and for neural architecture search in general, however, as discussed in Section 2.4.2 is the long training time of these models, which makes it computationally infeasible for a plethora of problems (Real et al. 2019).

Based on the promising results of Charte et al. (2020)'s approach in a range of computer vision problems (e.g., LeCun et al. (1999), Frankle and Carbin (2018)), first ways of compressing images are explored while minimising the loss of information needed for computer vision tasks such as classification and object detection. A prominent angle is to use convolutional autoencoders (discussed in Section 2.1). A challenge with convolutional autoencoders, however, is the complexity of their design, which involves a large number of hyperparameters that must be carefully chosen and tuned in order to achieve good performance on a given task. This challenge prevents inexperienced researchers from using these models effectively and can hinder the transferability of trained models to new problems.

Here, a novel neural architecture search approach is proposed based on neuroevolution to approximate the Pareto-front of convolutional autoencoders to overcome these challenges. This method is designed to optimise the trade-off between reconstruction loss and image compression. The image compression is calculated based on the size of the bottleneck layer, which is a key parameter in determining the model's overall performance.

The chapter continues with a preliminary study (Section 3.1) motivating the development of MONCAE, which reveals that pre-training with autoencoders can improve the performance of derived architectures. What follows is a detailed explanation of how MONCAE automatically discovers convolutional autoencoders in 3 different datasets (Sections 3.2 and 3.3). MONCAE achieves multi-objective optimisation by using the Hypervolume indicator. Next, the results are presented (Section 3.5) where the benefits from the multi-objective optimisation become apparent as MONCAE manages to beat the baselines in terms of balancing task performance and compression while taking several orders of magnitude less time than competitive approaches (ex. 93 GPU minutes compared to 1440 for MNIST). This is followed by a discussion and conclusion (Section 3.6) highlighting the implicit strengths of MONCAE, such as its ability to automatically adjust to the complexity of a given dataset and balance the importance of the objectives. A summarised version of MONCAE has been presented at the "2nd Workshop on Neural Architecture Search at ICLR 2021" Dimanov et al. (2021).

3.1 Motivation study

As discussed in Chapter 2, autoencoders enable researchers to achieve a variety of tasks, including dimensionality reduction (Wang et al. 2014), image denoising (Gondara 2016) and many others. More importantly, in the context of this work, they can also be used for image compression (Theis et al. 2017) and feature extraction (Goodfellow et al. 2016b), which is why a NAS for autoencoders is created in this chapter.

One of the key motivations behind MONCAE is that when a particularly effective convolutional autoencoder is discovered for a dataset (based on low reconstruction loss), this implicitly means that the encoding from which the images were reconstructed contains the information needed to complete the specified task (Goodfellow et al. 2016a). Consequently, encodings created by the bottleneck layers can be potentially used by a classifier or detector, which can complete the task at hand using only these encodings as inputs rather than the whole image.

Therefore, the process would achieve extra byproduct objectives in addition to the actual discovery of the autoencoder. The main three of these additional implications would be:

1. Splitting the feature extraction and classification tasks - This would allow the encoder part to conduct the feature extraction first. Then, the whole network can be reconstructed by adding the layers designed for classifying these features.
2. Performing unsupervised learning when the dataset labels are unavailable.
3. Facilitate transfer learning on different tasks for the same dataset using the extracted features.

In short, it can further foster already well-established training and machine learning pipelines of procedures.

The first implication would mean that researchers can split the feature extraction and problem-solving tasks during training, hyperparameter search or model search experimentation. This task separation would render an easier verification of results and latent representations comparison, which have traditionally been considered to be part of a 'black-box' process (Guidotti et al. 2020) In addition, if the reconstruction error is low, the latent representations can be used as a separate objective with an auxiliary output when the model is trained. This reconfiguration would allow for more fine-grained control of how the model is learning and can be easily achieved by the addition of auxiliary outputs and losses to already existing models (Goodfellow et al. 2016b)

Moreover, a series of challenges can be alleviated by adding an extra objective, such as selecting the right loss for training on some tasks. For example, problems with a significant imbalance present can incorporate an objective to foster class imbalance invariant reconstructions based on separate class-specific metrics.

3.1.1 Rationale

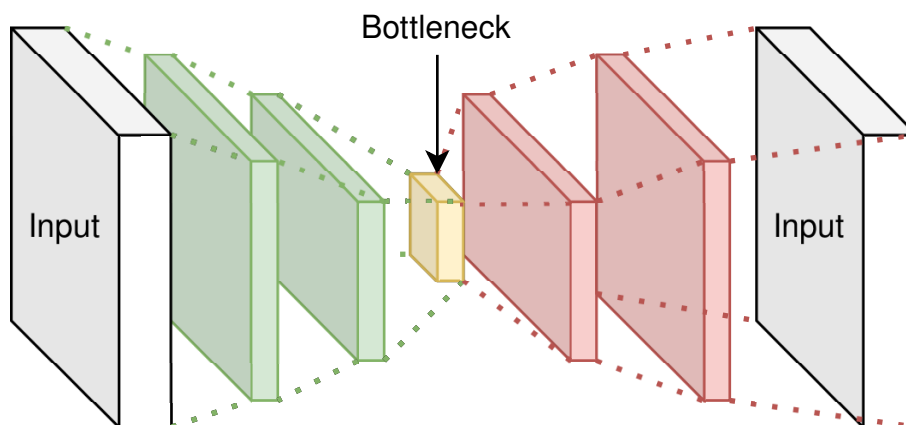
Training machine learning models is time-consuming and requires careful data collection and labelling. The availability of labelled data is often a limiting factor in the design and optimisation of model architectures for supervised learning tasks. As a result, the selection of model architectures must usually be made before the completion of the data labelling process, but the data is a prerequisite for the search. This prerequisite can result in suboptimal performance and a need for additional training and optimisation once the labelled data is available (Cunningham et al. 2008).

The rapid pace of development in machine learning often leads to rushed architecture search and model optimisation processes, which can be constrained by time limitations (Munappy et al. 2019). This rush can compromise the overall performance of the resulting model. It may also lead researchers or practitioners to neglect important aspects of the system, such as obscure metrics or insufficient testing (Goodfellow et al. 2016b). As a result, it is vital to carefully consider the various stages of the model development process and ensure that best practices are followed to produce high-quality models.

Using a convolutional autoencoder (example visualisation of the general structure in Figure 3.1) and optimising its reconstruction loss is not dependent on any labels, which can allow for the use of a well-working autoencoder to compress and discover the features of the dataset prior to the data labelling.

This unsupervised approach can improve the overall quality of the process and help with early debugging, proving to be of paramount importance (Myers et al. 2011). Moreover, the effect of data augmentation and preprocessing can also be established earlier.

Figure 3.1: General structure of symmetrical convolutional autoencoder



In addition to all that, if the feature extractor is discovered in advance, then the model search is confined to looking just for the last couple of layers to complete the task at hand instead of conducting neural architecture, hyperparameter or some other time-consuming search or process on the whole model at a much later stage (Goodfellow et al. 2016b).

The research questions when approaching this problem are formulated as follows:

1. Can representation learning be done in an unsupervised or semi-supervised fashion using convolutional autoencoders?
2. Can the results discovered with unsupervised representation learning be transferable to classification and other computer vision problems?
3. Is it more efficient to use conventional convolutional neural network classifiers, or can one learn the representations in unsupervised learning and then train only the classification layers?
4. What impact does training for unsupervised representations and then classification training have on the overall classification performance?

Based on the research questions, a hypothesis is proposed:

Hypothesis 1 *Feature extraction can be done before labelling a dataset. When the labels are made available, the learnt features can be used to discover or apply transfer learning to a classifier with competitive performance to the same classifiers being discovered and trained in the conventional black-box method.*

Null Hypothesis 1 *Conventionally discovered and trained convolutional classifier significantly outperforms both a classifier trained only on the discovered features from a convolutional autoencoder and a classifier which uses the discovered features to conduct transfer learning.*

Alternative Hypothesis 1 *Conventionally discovered and trained convolutional classifier significantly outperforms a classifier trained only on the discovered features from a convolutional autoencoder but fails to outperform significantly a classifier which uses the discovered features to conduct transfer learning.*

Alternative Hypothesis 2 *Conventionally discovered and trained convolutional classifier fails to significantly outperform a classifier trained only on the discovered features from a convolutional autoencoder but outperforms significantly a classifier which uses the discovered features to conduct transfer learning.*

Alternative Hypothesis 3 *Conventionally discovered and trained convolutional classifier fails to significantly outperform both a classifier trained only on the discovered features from a convolutional autoencoder and a classifier which uses the discovered features to conduct transfer learning.*

3.1.2 Methodology for testing Rationale

An experiment is designed to test this hypothesis using three different convolutional architectures across ten random seeds, and for each run, three separate experimental arms

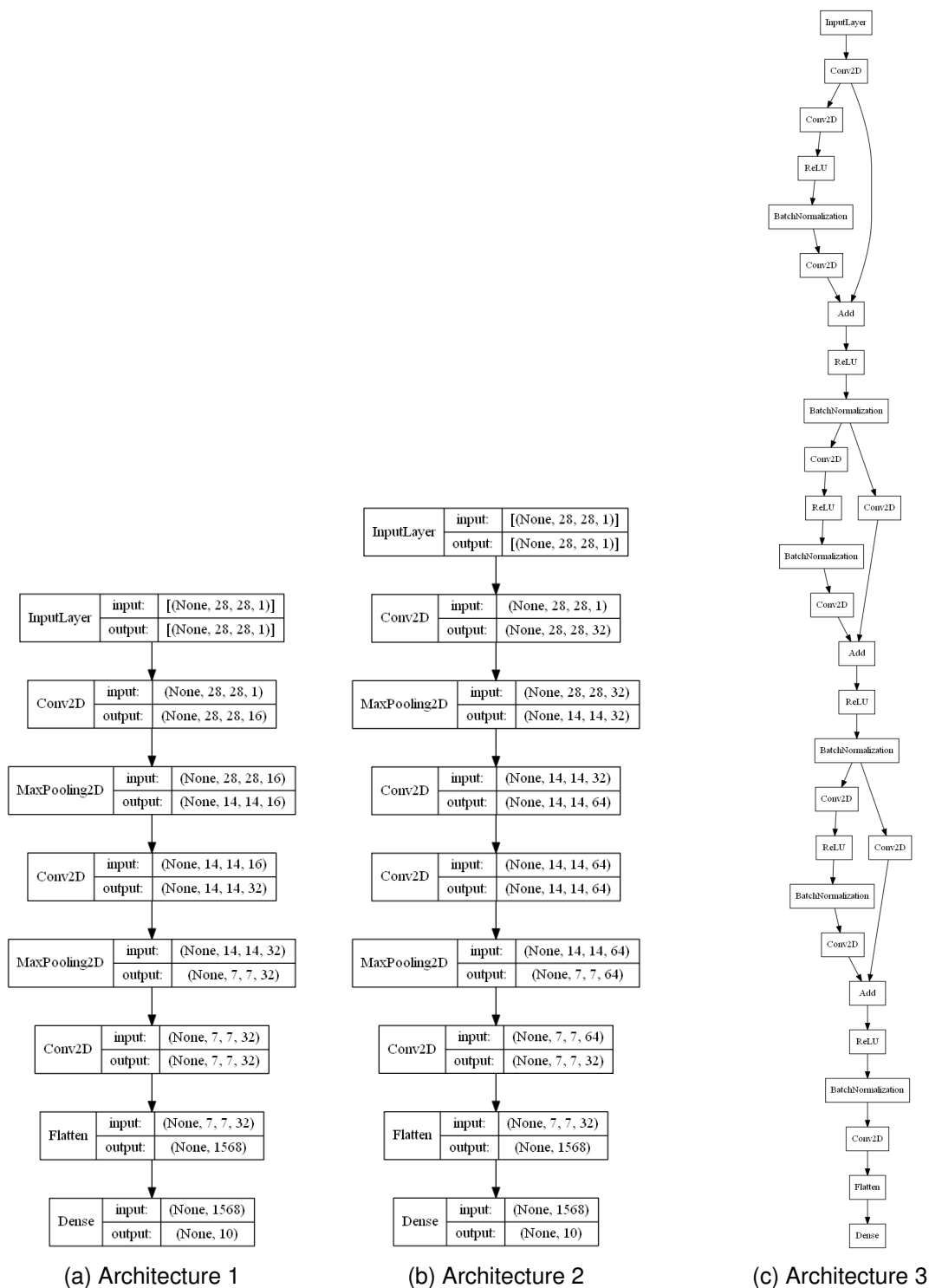


Figure 3.2: The three used architectures in the "motivation experiment". Architecture a) is a simple convolutional autoencoder with only three layers before the encoding, b) has four layers before the encoding as well as more than double the feature maps available to a), and c) is significantly more complicated with having three resnet blocks (equivalent to 15 layers as counted for the other architectures) before the encoding.

are created. The three architectures are conventional neural architectures (displayed in Figure 3.2) which have different depths and are made of different layers. Architecture 1 and 2 are similar but use different filter sizes in the convolutional layers. In contrast, architecture 3 is substantially more different from the other by having ResNet blocks (discussed in E). The dataset used for this experiment is MNIST. The control arm is a conventional sequential architecture constructed from feature-extracting convolutional and max-pooling or strided convolutions followed by a couple of layers that classify the features (referred to this model as the 'classification model'). The classification models are displayed in Figure 3.2. The **first experimental arm** uses the feature extraction layers as the encoder part of an autoencoder. Then the decoder is built with the same layers where the max-pooling layers are substituted for upsampling ones, and the strided convolutional layers are substituted for separable convolutions (covered in detail in Chapter 2). Then, an auxiliary model takes as input the output of the autoencoder by the bottleneck layer and adds the same classification layers as the control arm to produce what is referred to as the 'latent model'. The third model (**Experimental Arm 2**) is the same as the 'classification model' from the control arm. However, instead of training it from scratch, the weights from the trained autoencoder described above are used as initial weights before retraining ('transfer learning model').

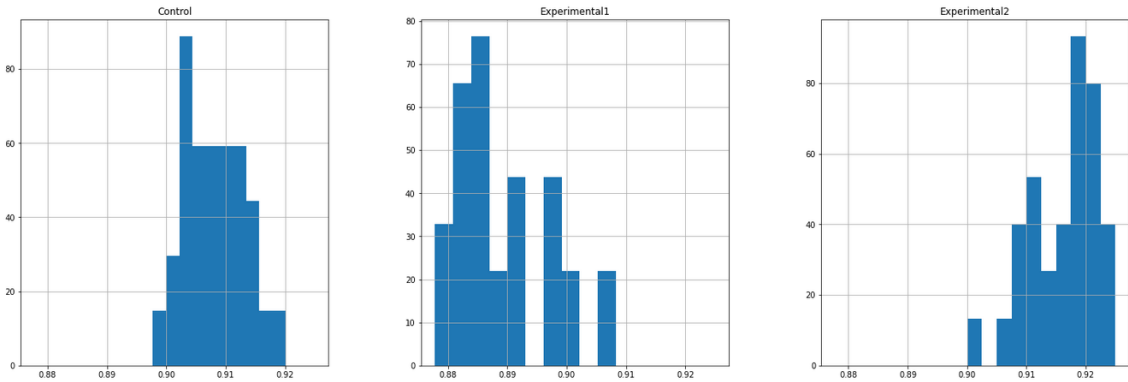
3.1.3 Experimental setup for testing Rationale

The batch size and epochs for all experiments are 64 and 100, respectively. These hyperparameters were chosen after some preliminary tests determined that with them fair comparison between the approaches can be achieved and also it was the maximum feasible batch size for the hardware the experiments were conducted with. The learning rate is specified to be 0.001 which is suggested by multiple works to be a good starting learning rate for the optimiser used throughout the experiments (especially for this datasets) (Smith 2018, Chrabaszcz et al. 2017, Liu et al. 2019c) with an additional callback, which acts as a dynamic learning rate scheduler and reduces the learning rate when there has not been any improvement of the value of the loss function for 2 epochs, in which case it multiplies the learning rate by a factor of 0.5. This choice is made to avoid overfitting and allow for fine-grained modifications by the optimiser (Adam) and is popularly used in the form of cyclic learning rate with similar rule sets (Smith 2017, Rodellar et al. 2022, Wang et al. 2023).

3.1.4 Results of testing Rationale

Figures 3.3 and 3.4 illustrate the results using these settings. A trend becomes apparent from the results depicted by the box plots of the categorical accuracies. The experiments on the Control arm seem to be better than Experimental arm 1 yet worse than Experi-

Figure 3.3: Distribution of accuracies for each experimental arm



Experimental Arm	$s^2 + k^2$	P-value
Control	0.792	0.673
Experimental 1	3.44	0.179
Experimental 2	3.17	0.204

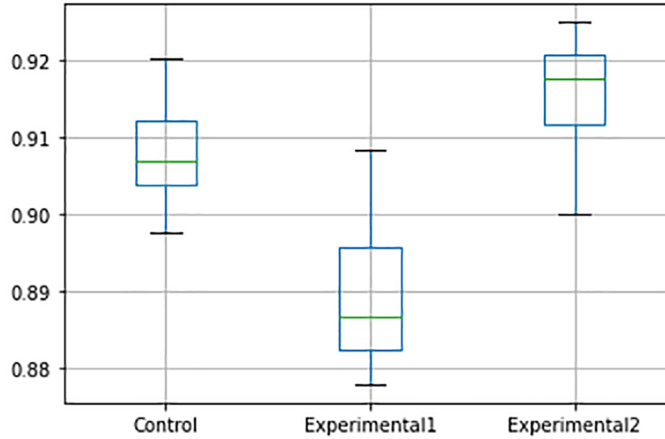
Table 3.1: Normality test results for the three experimental arms, where s is the skew test, and k is the kurtosis test. The test operates under the assumption that the null hypothesis is that the data comes from a normal distribution and anything above 0.05 p-value is considered high enough to reject this hypothesis.

mental arm 2. This finding suggests a limited value of the autoencoder’s representations since the latent representations-based classifier with the auxiliary added classification layers did not perform as well as the control arm. On the other hand, the illustration provides evidence that the transfer learning approach in the second experimental arm might contribute to an overall improvement of results.

Next, the hypothesis is further investigated using a normality test based on d’Agostino (1971), and D’AGOSTINO and Pearson (1973) to evaluate the probability distribution of each experimental arm being Gaussian (above $p=0.05$ threshold, the normality hypothesis is considered rejected). Based on the results in Table 3.1 and Figure 3.3, it becomes evident that none of the experimental arms follows a normal distribution. Thus, the statistical significance between the sets cannot be accurately determined using statistical tools that have a prerequisite for normality. Hence, the Mann-Whitney U rank test is used to determine if the results for each arm dominate the other.

The Mann-Whitney U rank test fails to conclude that the first experimental arm results are greater than the control arm, signified by a score of $U = 147.5$ and a p-value of 3.9996×10^{-6} , which strongly indicates that the control arm outperforms the first experimental arm (the encoder). Thus, alternative hypothesis 2 and alternative hypothesis 3 have to be rejected. Interestingly, when the same test is performed for the control arm and the second experimental arm, the Mann-Whitney score of 867 signifies an even

Figure 3.4: Box plot of performance across the three experimental arms for motivation test experiment for MONCAE. Notice how the Control arm does not outperform the second experimental arm but seems to outperform the first experimental arm. This finding can be attributed to the fact that both the decoder and the encoder are optimised during training, but the decoder is subsequently discarded.



greater gap between the scores. However, the p-value denoting the probability of the second experimental arm being better than the control arm is 0.9999999997, which means that the probability of the control arm being better than the second experimental arm is 3.69×10^{-10} . These results positively confirm that the second control arm outperforms the control arm, which renders us unable to reject the first alternative hypothesis but is enough to reject the null hypothesis.

A limitation of this preliminary study is that only one dataset (MNIST) is used in the experiments. Thus, some of the findings might capture spurious correlations or be data-specific, and more robust tests are required to explore if this hypothesis stands with other datasets. While this limitation affects MONCAE, the results in Section 3.5 connote that this hypothesis stands for Fashion MNIST and CIFAR-10.

3.1.5 Why is MONCAE important?

In short, this motivation study proves that having a well-working autoencoder is not only important purely for reconstruction or denoising samples but the architecture of the autoencoder, and the learnt information can be transferred when the data is used to address a multitude of problems. This finding opens up a door for a whole new methodology of designing training cycles, architectures and algorithms for deep convolutional learning.

These new procedures can benefit from a preliminary unsupervised learning training step, which can help researchers and practitioners debug their models and overall methodology early on but also help boost the final performance of the produced models. Chapter 5 explores such an idea where an autoencoder built with MONCAE is used for coreset discovery.

3.2 MONCAE Methodology

Let's now turn back to the automatic design of autoencoders. Several studies have attempted employing neuroevolution to find optimal autoencoders before (Charte et al. 2020, Sereno 2018, Alvernaz and Togelius 2017).

Recently, researchers have demonstrated that one promising approach is to split the problem into two stages: first, feature engineering and then using the extracted features to solve the problem at hand (Sereno 2018). To achieve that, Sereno (2018), Charte et al. (2020) and Alvernaz and Togelius (2017) have showcased the possibility of using an automatically discovered autoencoder to tackle this.

The idea is that the produced autoencoder can compress the inputs by up to a factor of 10 with just one full forward pass and still preserve enough information to achieve satisfactory classification performance above 97% in the case of MNIST. Rather than using the raw input for training or network discovery, encoding the inputs can be used instead. As shown in the motivation experiment, the present transfer learning approach appears to be more effective than using the encodings as inputs. Therefore, the transfer learning approach is used in the experiments when finetuning to the specified task.

Approaches like Charte et al. (2020)'s EvoAAA mainly focused on evolving a simple neural network with dense activation layers. Based on the results presented in these works, the idea of using neuroevolution to optimise autoencoders looks promising. However, the problem with the current methods is their limitation of not using convolutional and pooling layers, which is typically a powerful feature to consider (Gu et al. 2018).

In MONCAE, a Neuroevolution algorithm is built, roughly based on the principles of DEvol (Davison 2017), a neuroevolution algorithm for neural architecture search intended for image classification.

While retaining the basic working principle of the initialisation and variation, the search space and the encoding are designed to allow for neural architecture search of convolutional autoencoders.

Multi-objective optimisation through the use of the Hypervolume indicator discussed above is added. To the best of the authors' knowledge, this work is the first time in which **neuroevolution multi-objective optimisation has been used for automating the design of convolutional autoencoder architectures**.

To make any neuroevolution approach compatible with convolutional autoencoders, this chapter first examines how convolutional autoencoders are structured. Convolutional autoencoders usually have symmetrical encoder and decoder parts, in which the input is compressed and reconstructed respectively (Goodfellow et al. 2016b). The motivation behind them stays the same as non-convolutional autoencoders (discussed in Section 2.1). However, the big-picture goal is to make use of convolutions to compress big-image datasets into more manageable and smaller representations.

The underlying rationale of this approach is that if the decoder part of the network can successfully reconstruct the images to a satisfactory extent, then the information needed to complete the task should be present in the latent representations tested in the previous section.

The two main objectives of MONCAE are to a) minimise the reconstruction loss while b) compressing the latent representation vector as much as possible. In doing so, a third objective is added to monitor the complexity of the model. This way, a preference for models with a lower level of complexity is added.

The implementation of MONCAE ¹ uses Devol (Davison 2017) as a foundation, and the modular design allows for the heavy experimentation with each different operator in the algorithm. To add multi-objective optimisation, MONCAE utilises the hypervolume indicator (see below) and adjusts the encoding and decoding operators together with the evaluation process.

The selection process in MONCAE is of tournament (TS) type, that is, $TS(P) = \{p_1, p_2, \dots, p_n\}$ where $P = \{p_1, p_2, \dots, p_m\}$, p is an individual of the population P , and n is the desired number of individuals to be selected. The tournament selection iteratively chooses a pair of individuals, and based on their performance, one is kept and the other discarded. For the specifics of each operation, please consult the code ¹.

A severe limitation of AutoML approaches (and neuroevolution in particular) is that these algorithms take a tremendous amount of time to discover optimal solutions, making it infeasible for many modern-day scenarios (Lebedev and Lempitsky 2018, He et al. 2021, Kelly et al. 2023). . There are many different attempts to solve this problem, focusing both on the 'data space' (Wang et al. 2018, Singh and Lee 2017a) and the 'algorithmic space' (Hinton et al. 2015, Frankle and Carbin 2018).

MONCAE follows a similar methodology to Charte et al. (2020), but the neuroevolution algorithm is allowed to search for convolutional and pooling layers instead of using dense layers for the architecture. The search space also includes the depth of the network rather than simply the width of a single layer. Moreover, the hypervolume indicator (Rostami and Neri 2016, Guerreiro et al. 2020) (see details in Equation 3.2 below) is used to find a trade-off between two main objectives: 1) the reconstruction loss and 2) the level of compression, a metric reminiscent of the classic Bayesian information criterion (Watanabe 2013), defined as

$$\mathcal{L}(\mathbf{k}) = \log_{10} \prod_{i=0}^d (\mathbf{k}_i), \quad (3.1)$$

where \mathbf{k} is the vector containing the size of each dimension (k_i) of the linear map produced by the bottleneck layer of the convolutional autoencoder. The logarithm balances the different components of the objective function, ensuring they are all effectively con-

¹<https://github.com/DanielDimanov/MONCAE>

sidered in the optimisation process.

As mentioned earlier, the main instrument for the selection in MONCAE is the Hyper-volume ($\mathcal{H}v$) indicator, which is a well-established performance in multi-objective problems. The $\mathcal{H}v$ indicator measures the volume of a particularly relevant region contained in the m -dimensional space (m is the number of objectives), spanned by the objective functions f_1, \dots, f_m (namely, the region "covered" or "dominated" by the population of solutions $\mathcal{X} \equiv \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ with respect to the reference point \mathbf{f}^{ref}). It is typically defined as (Guerreiro et al. 2020, Zitzler et al. 2007):

$$\mathcal{H}v(\mathcal{X}, \mathbf{f}^{ref}) = \Lambda \left(\bigcup_{\forall \mathbf{x}_i \in \mathcal{X}} \left([f_1(\mathbf{x}_i), f_1^{ref}] \times \dots \times [f_m(\mathbf{x}_i), f_m^{ref}] \right) \right), \quad (3.2)$$

where \mathcal{X} contains all performance metrics for the whole population of solutions, \mathbf{f}^{ref} is the reference vector of entries $f_1^{ref}, \dots, f_m^{ref}$, the expressions in brackets $[f_i, f_i^{ref}]$ indicate the length lower-bounded by f_i and upper-bounded by f_i^{ref} , and Λ is the Lebesgue measure over the union of all the m -dimensional hyper-cube volumes (that is, the net volume in euclidean space).

Thus, and remarkably, further objectives (beyond compression level and reconstruction loss) can be seamlessly added in MONCAE.

Based on the multiple objectives (and pre-specified reference points for each one), first, the $\mathcal{H}v$ of the population at each generation is calculated and then evaluated through the contribution of each individual solution \mathbf{x}_i to the overall $\mathcal{H}v$ which is termed the contributing hyper-volume indicator, $c\mathcal{H}v$ (aka incremental $\mathcal{H}v$ indicator, $\delta\mathcal{H}v$) which is used as the fitness score,

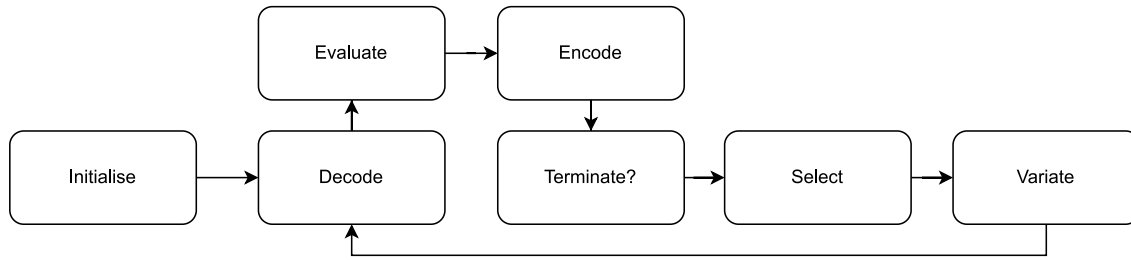
$$c\mathcal{H}v \equiv \delta\mathcal{H}v(\mathbf{x}_i, \mathcal{X}) = \mathcal{H}v(\mathcal{X} \cup \{\mathbf{x}_i\}) - \mathcal{H}v(\mathcal{X} \setminus \mathbf{x}_i). \quad (3.3)$$

The working principle behind $c\mathcal{H}v$ is that individuals \mathbf{x}_i from the optimal front of solutions are removed one by one from the approximation set. Then the $c\mathcal{H}v$ is recorded for each individual solution, which captures its contribution to the overall $\mathcal{H}v$. The process of MONCAE follows the general Neuroevolution cycle, and details over each step are discussed next.

3.3 MONCAE algorithm

First, some hyperparameters are set, namely the number of generations, the population size, the maximum number of convolutional (and pooling) layers, the maximum filter size in a convolutional layer and the maximum epochs (used when each architecture is evaluated), determining the genome size. The general steps of the algorithm follow the neuroevolution process depicted in Figure 3.5. In short, when the algorithm starts the

Figure 3.5: High-level flowchart of neuroevolution with all steps from general evolutionary algorithms with Decode and Encode operators added. The Variation operator can be further split into Crossover and Mutation in the case of Genetic algorithms.



evolutionary process, all genomes are of known size, making it easier to reference and allowing for Atavism (Rostami and Neri 2016) to occur. The idea of Atavism is that some of the layers might be disabled. So, some of this layer’s properties (e.g., the kernel size, the number of filters, etc.) will not be reflected in the phenotype (the actual neural network). However, this genome continues to carry genetic information about these layers. They become the equivalent of recessive genes in biology (Deb 2001b).

Table 3.2: MONCAE search space. Notice that these presets can be treated as high-level hyperparameters and can be easily modified to construct new search spaces.

Parameters	Range	Description
Active	{0,1}	if the layer is active or not
Num filters	$\left\{ \sum_i^{\log_2 max\ filters} 2^i \right\}$	number of filters
Kernel	{3,5,7}	kernel size
BN	{0,1}	if batch normalisation should be applied
Activation	$a \in \left\{ \begin{array}{l} 'relu' \\ 'sigmoid' \end{array} \right\}$	what activation function should be used
Dropout	{0,1}	should there be dropout after the layer after this one
Pooling	{0,1,2}	if max-pooling should be included

Following, the search space is defined. It comprises of all possible combinations of chromosomes in the genome (Table 3.2 outlines the possible values for each search parameter per layer).

Every single layer may be composed of convolution, max pooling, activation and dropout operation. Compound layers are also allowed, consisting of more than one of these operations. The only operation required is the convolution, which can also be disabled, which in turn disables the rest of the operations in the given layer too. This mechanism is triggered by zeroing the *Active* parameter. The total amount of parameters is thus calculated by the number of parameters per layer times the maximum number of layers + 1. The last added parameter determines the used optimiser during the training of the networks. Four optimisers are included in the search space, but they can easily be increased or decreased based on the given use case: 'Adam', 'Adagrad', 'RMSProp' and

'Adadelta'.

For the **initialisation** (step 1), a random population is generated (of the specified size). It is important to note that other strategies, like starting at the extremes or using state-of-the-art architectures as a starting point, might yield even better results in practice, but towards a better experimental control, here, random initialisation is used. When constructing the genomes, each parameter's position matters, and based on the position in the genome, the chosen parameter has a different range of possible values (as listed in the example search space of layer parameters in Table 3.2).

The **encoding** (step 2) of the genome is of length $(2 \times f \times k \times 2 \times \hat{a} \times 2 \times 3)^l + 1$ where f is the number of filters available based on the specified maximum number of filters, \hat{a} is the number of available activation functions used, k is the number of kernel sizes specified, and l is the maximum number of layers allowed.

After the population is initialised, the individuals are decoded into their phenotypes (CNNs). The approach follows a conventional autoencoder design where the inputs are first compressed using an encoder and then reconstructed from the latent space using a decoder (Alvernaz and Togelius 2017).

The symmetrical nature of the autoencoder structure greatly enhances the approach's search strategy by automatically inferring the decoder from the constructed encoder. So, this work only encodes the first part of the autoencoder responsible for compressing the information. Then a series of convolutional and upsampling layers are used to restore the original shape of the inputs and construct the decoder.

To achieve this, each layer from the encoder (the green and yellow layers from Figure 3.6) is constructed using the corresponding properties, and then for the decoder (the red layers in Figure 3.6) the layers are added in reverse order by substituting the pooling layers for upsampling ones. The compressed latent space is scaled up until the initial input shape is restored to decode the upsampling part of the network.

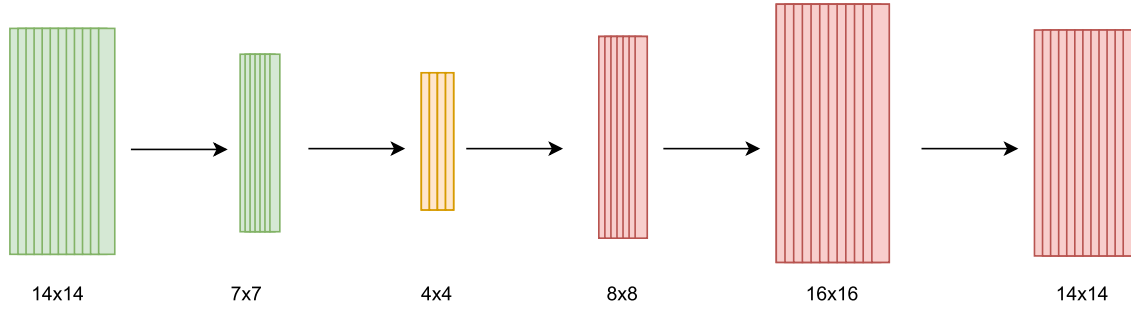
The last dimension of the output vector is the number of initial channels. This property will later allow for the use of the number of classes instead of colour channels, allowing the allocation of a single neuron per class in the last layer. The output of a MONCAE model is tasked to recreate the input from the highly compressed latent space. The decompression is achieved through upsampling layers, but they can be substituted for transposed convolutions also (Pons et al. 2021).

Achieving this requires the algorithm to iterate over the encoded convolutional layers again but in reverse order and exchange the pooling layers for upsampling ones. This way, the standard encoder-decoder structure of typical autoencoders is achieved (Azarang et al. 2019). Exceptions can occur in two special cases. One of the exceptions happens at the very end of the network, where a layer that fixes the number of filters is added to equal the number of channels of the input.

The second one happens conditionally to a specific process during the construction

of the encoder. If either of the two first dimensions of the output mask has been rounded off during the encoding, the procedure depicted in Figure 3.6 takes place. The decoding of the first part of the network stays the same, but when the upsampling is done, a new convolutional layer is added to adjust the mirror side's shape based on the rounding's position.

Figure 3.6: Decoding process if feature map with odd height or width appears. The green(downscaling) and the yellow (bottleneck) layers are part of the encoding. The red(upsampling) layers are generated based on the shape of the bottleneck and input size.



In the example, the input is of square shape, so in other terms, the shape goes from 14 - 7 - 4 - 8 - 16 - 14. The pooling that happens when the shape is 7x7 causes the next shape to be 4x4. Upsampling by 2 of the 4x4 feature maps causes it to be 8x8, which then causes a shape mismatch down the line. This phenomenon can happen anytime a pooling operation occurs after the shape is not an even number. Feature maps' shapes are stored while constructing the encoder to alleviate this obstacle. Then, it is used to come up with a rule that the shape should increase following this formula:

$$e = \begin{cases} s \mapsto s/2 & O^i \in P \\ s & O^i \notin P \end{cases} \quad | \quad i \in \{0, 1 \dots n^l\}$$

Where n is the number of layers, O is the set of all operations in the encoder, P is the set of pooling operations, and s is the current shape of the feature maps initialised to be equal to the input shape k . e is used to store all feature map shapes. Then, upsampling layers are used to facilitate the 2x increase of the shape and an extra convolutional layer with valid padding and 3x3 kernel size to do the -2 subtraction, as shown in the procedure below: This trick gives the algorithm full freedom in constructing the encoder part. However, a hard limitation is set in place that prevents the model from pooling when the shape of the feature maps gets below 4x4 to prevent the algorithm from searching for architectures with excessive data loss due to compression.

In the experiments, the architectures are **evaluated** (step 3) by training for \hat{e} epochs. Through experimentation, a good value for \hat{e} is determined to be 5, which allows for a reasonable estimation of the performance while significantly reducing the needed time

Algorithm 1: Constructing decoder part. Logical brackets are coloured for legibility.

Data: e, n, s, k, O, P
Result: decoder operations
 $D \equiv \emptyset$
for $i = n-1$ **to** 0 **do**
 if $O_i \notin P$ **then**
 $D \leftarrow D \cup O_i$
 else
 $D \leftarrow D \cup \text{upsample}$
 $s \leftarrow s \times 2$
 end
 if $(s \times 2 \notin e) \wedge ((s-2) \times 2 \in e) \vee ((s \times 2 \notin e) \wedge ((s-2) \times 2 = k))$ **then**
 $D \leftarrow D \cup \text{conv}((3, 3), \text{valid})$
 $s \leftarrow s - 2$
 end
 $s \leftarrow 1$;
end

for training. While doing so, early stopping is used to avoid training models that are not improving and conserve some resources for the rest of the process.

Then, results are verified by running the model on a pre-defined validation set that all individuals use. Here, various metrics are tried to report the fitness of each individual. The aim is to minimise the latent vectors in the bottleneck and the reconstruction loss. Hence, a multi-objective metric such as the $c\mathcal{H}v$ was required.

As mentioned, the first objective is the reconstruction loss for which binary cross-entropy between the output of the network and the input is used (as discussed in Chapter 2): $H(p, q) = - \sum_{x \in \{0,1\}} p(x) \log q(x)$. In this case, $p(x)$ would denote the label for x and $q(x)$ would denote the output of a model q given input x .

The next step, (**Terminate**), checks if the termination criterion is met. If it is, then the algorithm is stopped (in the context of MONCAE, this criterion is the number of generations).

The crossover is a one-point crossover, meaning a single point is randomly chosen. The offspring of the selected individuals is generated by taking the first part of one of the parents and the second part of the other one with respect to the chosen point for crossover. In terms of mutation, a criterion roughly based on simulated annealing (Van Laarhoven and Aarts 1987) decides when to apply a random chance of a random parameter.

For the change of parameters, a valid random gene is chosen and substituted with a new random one from the search space. The mutation is similar to simulated annealing in that at the start of the neuroevolution algorithm, the process starts with a higher mutation rate and decreases with the passing of generations.

Using multi-objective optimisation with a population-based approach such as neu-

roevolution allows for the examination of the whole final approximation set of produced solutions instead of being presented with one. Stakeholders are thus enabled to choose the model that best fits the context of the problem at hand.

3.4 Experimental setup

In these experiments, the reference points are set to **4 and 12** for the error and level of compression correspondingly. During development of the method various different reference points have been tried and they this hyperparameter tends to be sensitive to the desired output. Based on the determined ranges, these references points provide the best balanced solutions. If the reference point for level of compression is pushed more it starts generating smaller model but the sacrifice in performance is too big. On the other hand, if the model only focuses on the error it tends to overparameterise the produced network. Thus, by empirical analysis and to reflect the actual specified requirements in this work, 4 and 12 were chosen. Thus, the algorithm might be biased to pick models with better compression than ones with lower error. The process showcases how preference articulation can be easily added to the approach by setting different reference points for the hypervolume indicator.

Even though this is not the only way to add preference articulation in MONCAE, it is possibly one of the easiest. Based on the picked reference points, some objectives might be weighted differently (similar to how the loss minimisation is weighted more than the level of compression). In all training stages, if not otherwise specified, a batch size of 256 and 20 epochs is used. These two hyperparameters are tied to the hardware and available computation time. The batch size is picked to be the largest possible batch size to facilitate training on the available hardware, while the 20 epochs were chosen based on some influential works that suggest it as a good point for estimating overall performance (Bonet et al. 2021, Liang et al. 2019a, Emuoyibofarhe et al. 2020, White et al. 2023)









































The discovered architectures are then fine-tuned by adding 2 auxiliary layers of 200 and 10 nodes from the bottleneck. The choice for picking 10 nodes is dictated by the number of classes and the choice for 200 in the previous layer is motivated by early universal approximation theorem works such as Huang et al. (2006). Then, they are trained for 20 more epochs with the classification labels provided for the dataset at hand. The same process is followed when comparing to benchmarks.

MONCAE is evaluated on three datasets: MNIST (LeCun et al. 1999), Fashion-MNIST (Xiao et al. 2017) and CIFAR-10 (Krizhevsky et al. 2014). For each dataset, 10 runs are conducted with different random seeds to allow for some statistical significance for 20 generations (motivated not only by previous evolutionary computation success and potential diminishing returns after this point (Tan et al. 2009), but also from real-world evolution, where "mechanisms ... can influence rapid evolutionary change within 20 gen-

erations or fewer (Prentis et al. 2008)) and a population size of 20, a maximum of 20 epochs and using early stopping. After the process is complete, the final population is finetuned by training for an extra 20 epochs using the parameters, hyperparameters, optimiser and architectures discovered by the algorithm. Experiments were implemented in Python, TensorFlow 2.4 and a single RTX 3090 NVidia GPU with CUDA 11.1.

CIFAR10 (Krizhevsky et al. 2014) is a dataset composed of 10 classes: aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Usually, the dataset is used as-is, and the task for the dataset is image classification, where each image is classified as being a single class out of the ten. There are extensions of this task that group different classes together and redefine the classification to recognise the newly constructed group instead of the original label (Wan et al. 2020, Ma et al. 2021). For more information about any of the datasets mentioned, please refer to Appendix G. These three datasets are picked to make this work as comparable as possible to other similar state-of-the-art approaches and showcase its performance on something that the domain reader will be familiar with. Unfortunately, the three datasets do not represent the complexities of modern, more sophisticated data but simultaneously make it possible to compare to the most significant amount of state-of-the-art works. Moreover, and more importantly, this work introduces a novel approach that explores the feasibility of using multi-objective neuroevolution in the problem and lays a potential stepping stone for new approaches. Hence, the focus is on attempting to reject a null hypothesis with minimal possible experiments but still ensuring it wasn't purely lucky. For this reason, multiple widespread datasets are used, and the experiments are run numerous times with different seeds. Using CIFAR-10, in particular, is vital as most designed benchmarks for AutoML are designed, tested and compared on this specific dataset, and there has been a trend to use it with any new approaches to facilitate easier comparisons (Mehta et al. 2022, Siems et al. 2020, Ying et al. 2019). Moreover, the only other similar approach at the time of the development of MONCAE - EvoAA-Diff uses only these 3 vision datasets and using the same datasets allows this work to showcase MONCAE's competitive results.

Figure 3.7: A population of solutions for MNIST. A single digit is used as an example of the performance of every single model from a run picked randomly. For each model, an example of the reconstruction of a single digit can be seen, as well as the achieved reconstruction loss and level of compression.

Original										
Reconstructed										
Rec loss	0.115	0.191	0.169	0.161	0.161	0.083	0.150	0.156	0.144	0.174
Level of compression	1.806	1.204	1.204	1.204	1.204	2.107	1.204	1.204	1.806	1.204
Original										
Reconstructed										
Rec loss	0.079	0.079	0.069	0.070	0.154	0.074	0.061	0.071	0.095	0.079
Level of compression	2.292	2.292	2.292	2.292	2.292	2.292	2.894	2.292	1.806	2.292

3.5 Results

Turning now to the results, it is worth mentioning again that when interpreting the new objective (level of compression), it is subject minimisation and not maximisation, meaning that a **lower level of compression is better** than a higher one. The two objectives are also listed below to compare the different solutions within the population and the prediction of each model.

It can be observed that, while not all of the solutions lie on the Pareto-optimal front and some models are dominated by others, there are some models with higher resolution outputs and lower loss that have achieved higher levels of compression, as well as others with lower resolution and higher loss but still relatively good compression.

Averaging the results over the entire population would not accurately capture the goal of producing an approximation set from which to choose. Therefore, the results from the runs in which the latent representations compress the input by at least a factor of 2 are presented, as more accurate but larger representations are not the focus of this work, revealing yet another way that preference articulation can be achieved with MONCAE.

A more detailed breakdown of the results for each dataset is discussed next, but what is consistently apparent across all experiments is that the averaged results have high variability and are, therefore, unreliable for evaluating individual solutions' quality. Consequently, the best architecture based on the cHv is chosen within the optimal set of solutions and presented as MONCAE-s1.

Since this is the first work to create a NAS algorithm for convolutional autoencoders,

Table 3.3: MNIST results from 10 independent runs using a conventional autoencoder with variable bottleneck layer size, a simple convolutional autoencoder, one produced from EvoAA, the best one from MONCAE and the MONCAE average. The + symbol denotes that the method is manual, and that is why there is no score (-) for the total time. Notice that the autoencoder discovered by MONCAE with the highest cHv (MONCAE-s1) achieves the best classification loss and accuracy while compressing the data the most out of the whole selection. MONCAE-average results are produced by running MONCAE 10 separate times, and because MONCAE aims to produce Pareto optimal solutions with a high standard deviation (denoted by \pm).

Approach	Rec loss	Layers	Bottleneck	LOC	Cl loss	Cl acc	Total time w/o ft (min)
EvoAA-Diff	7%	1	162	2.21	12%	96.31%	1440
Autoencoder+	6%	1	392	2.59	11%	96.59%	-
Autoencoder+	7%	1	196	2.29	12%	96.38%	-
Conv Autoencoer+	8 %	3	14-14-2(392)	2.59	10%	96.90%	-
MONCAE-s1	9%	25	4-4-4 (64)	1.81	9%	97.29%	82
MONCAE-average	14.9 \pm 7.4%	22 \pm 12	83 \pm 91	1.92 \pm 1.95	39 \pm 50%	86.25 \pm 19.03%	93 \pm 22.1

the approach is compared to EVOAA(discussed above), an autoencoder compressing the input dimensions to half, another one that does the same but compressing the input to 25% and a shallow convolutional autoencoder suggested by some works to achieve good results (Zhang 2018, Cheng et al. 2018).

To boost the clarity and legibility of the tables, the standard deviation is omitted where it is below 0.5%, which turns out to be the case for all tested approaches except for the averaged MONCAE runs.

Starting with MNIST, what stands out in Table 3.3 is the discrepancy between the achieved reconstruction loss and the classification loss/accuracy. Interestingly, the simplest autoencoder achieves the best reconstruction loss, which can be attributed to its largest bottleneck layer. The bottleneck layer has allowed the model to keep enough parameters to reconstruct the 28x28 image well but has seemingly had an adverse effect on its discriminative power compared to the convolutional approaches judging by the classification results.

Strikingly, MONCAE’s best cHv architecture achieves the best classification accuracy (and loss) while compressing the input more than 10 \times , compared to 2 \times -4 \times with other approaches.

Figure 3.7 presents an image per model for one of the runs. Each set of two images is a random sample taken from the validation set compared to the result of running it through a model from one of the 20 models in the final population.

The figure illustrates a sample population from one of the runs, showcasing the variety of available models produced. The total time in the table is the time for the whole algorithm to do 1 run with the constraints specified above.

Next, this chapter focuses on Table 3.4 where the results for Fashion-MNIST are consistent with the ones for MNIST. Notice again that the MONCAE-s1 architecture achieves the best scores in terms of classification, but fails to beat the other approaches in terms

Table 3.4: Fashion-MNIST results from 10 independent runs following the same conventions as the MNIST table (3.3) presented above. The + symbol denotes that the method is manual, and that is why there is no score for the total time. Again, the autoencoder discovered by MONCAE with the highest cHv (MONCAE-s1) achieves the best classification loss and accuracy while compressing the data the most out of the whole selection. Similarly to the MNIST results, the standard deviation across the produced populations from the 10 runs is high.

Approach	Rec loss	Layers	Bottleneck	LOC	CI loss	CI acc	Total time w/o ft (min)
EvoAA-Diff	0.258	1	256	2.41	0.35	87.38	1440
Autoencoder +	0.255	1	392	2.59	0.35	87.49	-
Autoencoder +	0.26	1	196	2.29	0.35	87.28	-
Conv Autoencoer +	0.28	3	14-14-2(392)	2.59	0.36	87.10	-
MONCAE-s1	0.29	14	7-7-4(196)	2.29	0.34	87.56	844
MONCAE-average	0.27±0.04	18±8	166.92±168.9	1.64±0.4	0.40±0.24	85.13±9.42	91.2±23.1

of reconstruction loss.

An interesting finding is the close performance of the default autoencoders with heuristic rules and EvoAA’s models (Charte et al. 2020). Even compared to the loss results presented in their study, MONCAE still yields better results.

Notice that in the Fashion-MNIST experiments, the average bottleneck representation is relatively larger than the one for MNIST, regardless of the use of the same hyperparameters and the equal size of the inputs.

In contrast to the earlier findings in this study, however, what can be seen in Table 3.5 is that the classification task with CIFAR-10 is progressively more difficult, which is indicated by the significant decline in classification performance for all methods. In contrast to the other two datasets, in CIFAR-10, the compression achieved by MONCAE is only 3x. However, the significant fluctuation in classification loss and accuracy makes MONCAE clearly more favourable than the alternatives. Remarkably, EvoAA’s near 80× compression leads to a classification accuracy of 35%, which is still better than random, but the difference between the objective scores and the fact that there might be applications where 35% can be acceptable performance is precisely why MONCAE generates a front of solutions for the user to choose the right architecture for their purpose.

While MONCAE-s1 is based on the highest cHv , this only means that this architecture has achieved the best balance given the objectives and specified area of interest (through the reference points). However, this doesn’t mean MONCAE-s1 is the best architecture from the produced front of solutions for all purposes.

More broadly, the reconstruction and classification error is the lowest for MNIST. The pixel reconstruction loss and accuracy indeed leave room for improvement, but remarkably, these architectures were discovered using only 20 generations with a population size of 20 for around 1.5 GPU hours on average. Moreover, the autoencoders are trained for only 20 epochs, and while in the presented experiments, this is sufficient for the models to converge, it can be argued that further hyperparameter optimisation (especially learning rate scheduling), regularisation (e.g. l1 and l2) and data augmentation can also

Table 3.5: CIFAR-10 results from 10 independent runs following the same conventions as the MNIST table (3.3) presented above. The + symbol denotes that the method is manual, and that is why there is no score for the total time. Notice that this time EvoAA's autoencoder offers the best compression but is severely outperformed by the other baselines as well as the MONCAE models, which again manage to achieve the best classification accuracy.

Approach	Rec loss	Layers	Bottleneck	LOC	CI loss	CI acc	Total time w/o ft (min)
EvoAA-Diff	0.61	1	39	1.51	1.8	34.95%	1440
Autoencoder +	0.56	1	1536	3.19	1.48	47.56%	-
Autoencoder +	0.57	1	768	2.89	1.5	46.60%	-
Conv Autoencoer +	0.57	3	16-16-2(512)	2.71	1.52	46.36%	-
MONCAE-s1	0.57	21	8-8-64(1024)	3.01	1.09	62.20%	2457
MONCAE-average	57.84±0.03	22±7	702±797	2.85±2.9	1.6±0.25	43.5±9.35	123±26.1

potentially boost these results. A more detailed look at the outputs of MONCAE-s1 for MNIST is available in Figure 3.8. The same can be found for F-MNIST and CIFAR-10 in Figure 3.9 and 3.10, respectively.

Figure 3.8: MNIST autoencoder with bottleneck of 4x4x4 and reconstruction loss of 0.0817. Notice that the digits are almost indistinguishable from the original inputs despite being reconstructed from a 12 times smaller representation.

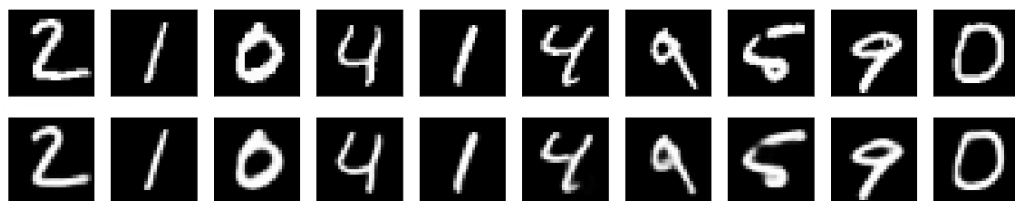
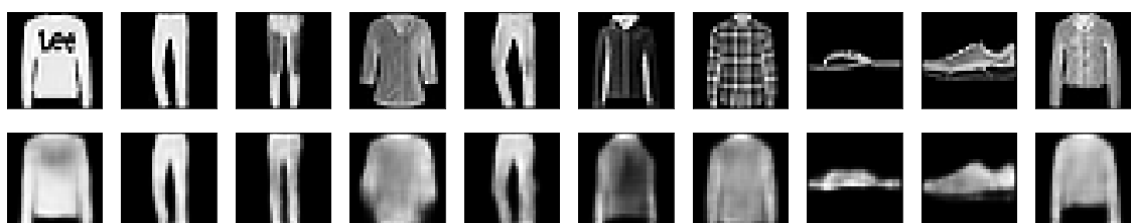
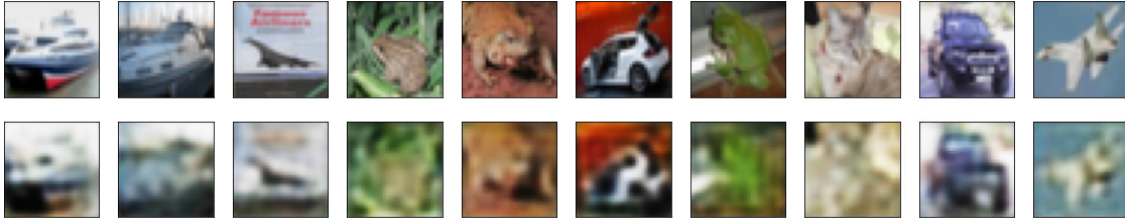


Figure 3.9: Fashion-MNIST autoencoder with bottleneck of 4x4x4 and reconstruction loss of 0.289. Notice that while some of the details are lost due to the high compression in the bottleneck layer, the different clothing articles are still easily recognisable, meaning that the salient information for classification is conserved in the 4x4x4 representation.



However, the fact that the algorithm managed to compress the dataset to representations that are (2x2x4) while still retaining enough information to reconstruct back the images correctly, with 50% of the data being accurately reconstructed, shows that the potential of the approach. With larger population sizes, more epochs, more generations and expanding search space, this approach can be scaled to larger datasets. Nevertheless, scaling the approach to multi-channel inputs remains a challenge (Figure 3.10).

Figure 3.10: CIFAR-10 autoencoder with bottleneck of $8 \times 8 \times 8$ and reconstruction loss of 0.564. In difference to the MNIST and Fashion-MNIST results, here the convolutional autoencoder attempts to capture more of the colours than the actual features or concepts of what makes up a class. Speculation can be made that this is because colour can be a "loud concept" (Kazhdan et al. 2021); thus, the loss function needs to consider this.



Figures 3.8, 3.9 and 3.10 present some of the best results obtained for all of the three datasets, showcasing the balance between compression and reconstruction loss. From the images, it can be seen that the MNIST reconstructions are hardly distinguishable from the originals. The results for F-MNIST are also still recognisable, but most of the details are lost during compression. Images for CIFAR-10, on the other hand, barely resemble the original images for the most part, which illustrates how adding the extra colour channels and complex objects increases the complexity of the problem and hence limits the extent to which a dataset can be compressed in lower-dimensional representations. Another plausible explanation is that this might be caused by the "loudness" of the concept of colour (Kazhdan et al. 2021). CIFAR-10 is also a fairly noisy dataset where the images have already been compressed to 32×32 pixels.

Noteworthy, some of the objects in the last two images of 3.10 look similar, and a general capture of the distribution of colours can be observed. This finding can mean that the cross-entropy loss might be more sensitive to colour distribution change than other important concepts (Kazhdan et al. 2020) being present in the reconstructions.

3.6 Conclusion and Future Work

This chapter presents a novel approach for automatically constructing convolutional autoencoders via neuroevolution architecture search methods to approximate the Pareto-front of solutions. While some of the results, especially for CIFAR-10, are still sub-optimal, this approach is designed to be a stepping stone towards automating the process and making it considerably faster.

The main goal here is to speed up AutoML and allow decision-makers to articulate preferences, offering them a set of models to choose from instead of just a single one. However, the chosen generations and population size (for computational cost reasons) might be still insufficient to thoroughly explore such an enormous search space.

In conclusion, MONCAE is an effective and efficient tool for searching for convolu-

tional autoencoders. By allowing users to specify different reference points and choosing from a set of generated solutions, MONCAE enables the selection of custom-tailored autoencoders that are well-suited to specific scenarios. The autoencoders generated by MONCAE can be used in a variety of generative models, including GANs, stable diffusion, and transformers.

Future work should explore scaling the algorithm for significantly larger datasets and expanding the search space available by adding new hyperparameters, as well as potentially achieving end-to-end neuroevolution. Moreover, given the popularity of skip-connections in recent computer vision state-of-the-art models, they can be added as part of the encoding and allow for an even more flexible search of architectures while expanding the approach to problems like semantic segmentation and object detection.

Chapter 4

RAMOSS - Resource-Aware Multi-Objective Semantic Segmentation

Recent advancements in Neural Architecture Search (NAS) enable the automatic discovery of neural architectures that are competitive with state-of-the-art, manually designed ones by experts (Guha et al. 2023, Kang et al. 2023, Elsken et al. 2019, Liu et al. 2021). This new paradigm has been revolutionary since it allows the discovery of architectures to be delegated to Automated Machine Learning (AutoML) approaches (He et al. 2021, Liu et al. 2021). In addition, the combination of multi-objective optimisation with NAS methods such as neuroevolution (Real et al. 2020, Lu et al. 2019) or reinforcement learning (Tan and Le 2019a) could potentially generate models that not only perform well but are also optimised to run faster or to comply with additional constraints (Lu et al. 2019).

The previous chapter discussed the use of neuroevolution for neural architecture search in autoencoders. As reported by Kang et al. (2023), Li et al. (2021) and He et al. (2021), the use of AutoML (automated machine learning) has gained significant attention in recent years as a means of streamlining the development process of machine learning models. Moreover, neuroevolution approaches for NAS enable the exploration of trade-offs between multiple objectives within the set of solutions provided by the algorithm (Real et al. 2020, Lu et al. 2019). However, a significant limitation that has hindered its widespread adoption is the time and computational resources required for its implementation (Abdelfattah et al. 2021). The MONCAE algorithm presented in the previous chapter demonstrated outstanding potential for accelerating the discovery of convolutional autoencoders for popular low-dimensional computer vision datasets. Also, it showed promise in reducing the time needed for the search. In addition, MONCAE showcases the importance of using multi-objective optimisation often neglected in this field (as discussed in Chapter 2). However, MONCAE has several limitations. Firstly, it

does not address semantic segmentation problems, which constitute a significant portion of current computer vision demand in industry (estimated to be around 40% by key industry companies, according to an interview with the CEO of Tenyks Limited). Secondly, it lacks the ability to encode skip connections, which have been crucial for the success of CNNs (as discussed in Chapter 2 and the appendix). Thirdly, it fails to exploit the progress of evolutionary computation algorithms. Finally, like most NAS approaches, it is designed for low-dimensional data, and additional heuristic methods would be required to scale it to high-dimensional datasets.

Semantic segmentation, which is vital for critical systems in domains like medicine (Dhamija et al. 2023, Khan 2023) and security (Zhang et al. 2023a) (discussed in more detail in Chapter 2), largely relies on manually-designed algorithms requiring significant domain expertise and computational resources. U-Net architectures are a predominant solution, manually constructed (Alsabhan et al. 2022) and possibly automatable through NAS (Weng et al. 2019). However, most NAS methodologies target image classification Xu et al. (2019), Liu et al. (2022a), Tsamardinos et al. (2022), with computational cost being a significant deterrent. Several strategies, including the use of a 'SuperNet' and weight-sharing, have been proposed to expedite the evaluation of candidate networks (Yu et al. 2021, Liu et al. 2022b, Xu et al. 2023). Recent efforts seek to bypass the intensive training process using pruning scores as performance proxies, although their applicability remains primarily confined to classification tasks (Abdelfattah et al. 2021).

As discussed in Chapter 2, neuroevolution, like other AutoML approaches, is mainly aimed at classification problems (with some exceptions, e.g. pose estimation (McNally et al. 2020) or object detection backbones (Operiano et al. 2020)). Hence, it is often unsuitable as a generic solution for other problems, particularly for semantic segmentation (Nagarajah and Poravi 2019). Here, the aim is to address this gap in the literature as well as the caveats of MONCAE by proposing a novel resource-aware, multi-objective neuroevolution-based NAS approach designed explicitly for semantic segmentation (RAMOSS). The contribution of this chapter is three-fold:

1. A novel encoding-decoding strategy for discovering optimal convolutional architectures in semantic segmentation problems is presented and evaluated.

2. One of the critical problems in AutoML, the high computational cost, is addressed. Towards this goal, a new sampling approach, progressive stratified sampling (PSS) for semantic segmentation, is presented. This method allows for the reducing the training time by several orders of magnitude while preserving the performance ranking of candidate architectures obtained from training with the entire dataset. Thus, this is at a much higher -suboptimal- computational cost. Overall, this sampling reduction approach renders an optimal computational cost-test error trade-off with respect to previous approaches and, hence, an efficient semantic segmentation.

3. Lastly, an open-source, multi-objective optimisation benchmarking framework based

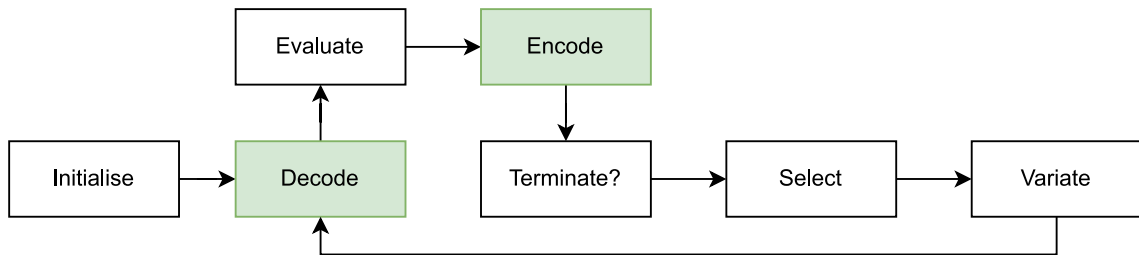
on the approach for testing NAS strategies on a wide range of computer vision tasks, is introduced.

Furthering the quest for more efficient and effective AutoML that can handle high-dimensional real-world datasets, this chapter endeavours to discover resource-aware architectures through neuroevolution capable of performing well on semantic segmentation tasks while also being efficient in terms of resource usage. To this end, a search space is designed that is able to capture the various connections between layers. This makes the search space contain 2.4×10^{64} possible architectures (compared to 4.4×10^{24} in MONCAE).

This is, to the best of the authors' knowledge, the first resource-aware multi-objective neuroevolutionary architecture search (NAS) method for semantic segmentation. First, one of the main caveats in NAS is addressed (their high computational cost) with a new technique termed progressive stratified sampling (PSS) in Section 4.1.3. This strategy is specifically devised to speed up the search by order of magnitude with a minimal sacrifice in performance. The performance of the proposed approach is demonstrated in a range of computer vision problems, including CIFAR10 and Cityscapes datasets. The choice for these two datasets is motivated by their popularity as established benchmarks in literature especially in the field of AutoML and NAS (Real et al. 2018, Liu et al. 2018, Lu et al. 2019, Zhou et al. 2020, Chen et al. 2014, Shaw et al. 2019). This choice allows RAMOSS to be easily compared with these state-of-the-art approaches without the need to redesign and rerun the approaches from scratch on newly chosen ones. Moreover, in the scope of this work, such extra experiments are rendered unfeasible because of the computational requirements of some of them. Also, using CIFAR-10 allows the work to be used and compared with NASBench and shows the general progression from Chapter 3 since it was also used there.

Next, the methodology of RAMOSS is discussed (Section 4.1) with a detailed explanation of how the encoding and decoding of the UNet architectures is done as well as the connections between layers. Next, the results are presented using some standard benchmarks and a high-dimensional dataset (Cityscapes) where RAMOSS generates state-of-the-art competitive architectures (76.3% mean IOU, which is $\pm 4\%$ compared to state-of-the-art expert-designed models) at a significantly lower computational cost (more than 1000 times faster than some of the alternatives with up to 30 times fewer parameters with up to 1.5% worse test error in CIFAR-10); and thus, it outperforms previous approaches in terms of balancing performance and resources (Section 4.3). The chapter finishes with some concluding remarks (Section 4.4) where RAMOSS is described as a stepping stone towards more general, efficient and effective multi-objective AutoML. A summarised version of this chapter has been published in UKCI 2022 (Dimanov et al. 2022), where it has gained the "UKCI 2022 Best Presentation Award".

Figure 4.1: Neuroevolution process stages. The main contributions are highlighted in the shaded green boxes.



4.1 Methodology

Here, a novel multi-objective neuroevolution approach for an effective Semantic Segmentation is presented, automatically providing UNet-based models in a resource-efficient fashion.

The approach is built with flexibility in mind by allowing full freedom of connections between layers and an option to search for different activation functions, different kernel sizes of convolutional layers, types of pooling and beyond. This relaxation results in a tremendously large search space, and thus, is shrunk by removing highly correlated *encodings* (see details in the next section). The *genome* is also compatible with multiple optimisation libraries such as Pymoo (Blank and Deb 2020) PyGMO¹ and Platypus^{2,3}. The architectural decisions allowed us to seamlessly compare different optimisation algorithms and provided further control to the experiments. The source code of the approach and all the scripts used to download, process, and load each dataset are publicly available and well-documented. Readers are encouraged to reproduce and improve upon the results⁴.

This work focuses on neuroevolution, hence refers to each encoded architecture as a **genome**, and the collection of architecture in one generation (in the case of evolutionary approaches) as **population**. The chapter follows the conventional neuroevolution skeleton process (Stanley et al. 2019b) illustrated by Figure 4.1, which highlights the area of the main contribution. In addition, it is interesting to note that the encoding approach can be adapted for optimisation with other methods, such as reinforcement learning.

4.1.1 Novel encoding and decoding approach

In order to encode an architecture, RAMOSS uses a flat list of values representing different properties for the layers of the network, similar to the approach followed in Devol⁵ and

¹<https://esa.github.io/pygmo/>

²<https://platypus.readthedocs.io/en/latest/>

³Due to limitations on some of the libraries, it is recommended to use the approach with Pymoo.

⁴<https://github.com/DanielDimanov/RAMOSS>

⁵<https://github.com/joeddav/devol>

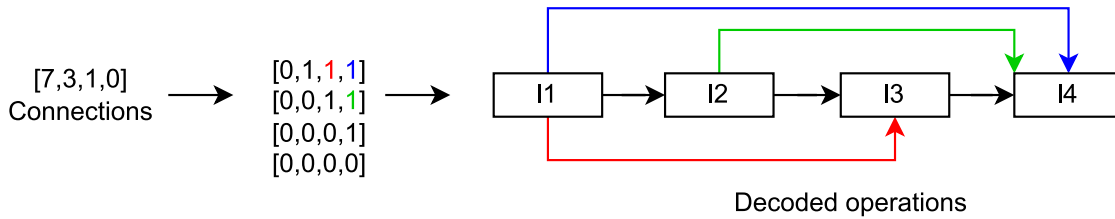
Table 4.1: Layer parameters defining the search space. n_c , n_d are the number of convolutional and dense layers, respectively (see main text).

Parameters	Range
Active	$\{0,1\}$
Num filters	$\{2^i\}_{i=1}^{\log_2 \max \text{ filters}}$
Num nodes	$\{2^i\}_{i=1}^{\log_2 \max \text{ nodes}}$
Kernel	$\{3,5,7\}$
Batch Normalisation	$\{0,1\}$
Dropout	$\{0,1\}$
Pooling	$\{0,1,2\}$
Connections	$\{i\}_{i=1}^{2^{(n_c+n_d)}}$

in Rostami (2014). For each convolutional layer, RAMOSS encodes all properties listed in Table 4.1 with the exception of the number of nodes. For the dense layers (used in the classification experiments), RAMOSS uses the following attributes: 'Active', 'Number of nodes', 'Activation' and 'Dropout'.

Table 4.1 also depicts the search space used in the experiments and illustrates the default ranges for each of these variables in the genome. The value of 'connections' for each layer in the encoding translates to a binary upper triangular matrix row. The matrix is (like in the NASBench (Ying et al. 2019) and DARTS (Liu et al. 2018) models) then used to construct the network connections (see Figure 4.2).

Figure 4.2: The process of decoding integer parameters for connections to decoded operations.



The start of the encoding of each layer is determined by each consecutive l^{th} property, where l is the layer size. In contrast to other encoding methods (Radiuk and Kutucu 2020), RAMOSS allows for macro and micro search of architectures simultaneously. As pointed out in (Radiuk and Kutucu 2020), an essential property of neural architecture search algorithms is that they do not only use repeatable structures but rather come up with more flexible ones, like in the present study. In this work, each architecture is encoded using $(n_c \times s_c) + (n_d \times s_d) + 1$ variables where n_c is the number of convolutional layers, s_c is the convolutional layer shape, n_d is the number of dense layers and n_d is the

dense layer shape.

For each dense layer(used only in classification problems), RAMOSS uses a similar approach but with fewer variables and do not allow skip connections. Finally, the optimiser used during the training stage is specified as a part of the evaluation co-routine.

While some of the parameters are straightforward, the parameter 'Active' (Table 4.1), which specifies if the layer should be ignored entirely during decoding, is critical for RAMOSS like in other algorithms (Lin et al. 2011, Shenfield and Rostami 2017). This parameter models a genetic mechanism, termed Atavism (Shenfield and Rostami 2017), which operates like recessive genes in biology. In the encoding, Atavism allows for specific layer parameters to be optimised while the layer is disabled and re-enabled in any succeeding generation.

Next, for decoding, the encoded genome is transmuted into a full-fledged TensorFlow model, which is evaluated by training for a given number of epochs. Notably, for autoencoders and semantic segmentation problems, RAMOSS stores only the first part of the architecture (the downsampling component) and then construct the upsampling part as a mirrored subnetwork of the former, in line with the state-of-the-art UNet-like models approach for these problems (Taghanaki et al. 2021). This is achieved by substituting the pooling layers in the encoder with corresponding upsampling layers, such that the output shape of the network is successfully adjusted. Thus, the output consists of a single final convolutional layer with the same number of filters as the number of classes to be discovered. However, two exceptional cases exist in which decoding deviates from the approach discussed so far. To address these exceptions, extra operations were used to adjust layers of incompatible shapes, as will be discussed in the next section.

Uneven filter map shape during the upsampling stage:

The first exception occurs when output mask dimensions are rounded off during the construction of the encoder. In this situation, the decoding of the encoder part of the network stays the same. However, a new convolutional layer is added after upsampling to adjust the shape on the mirrored side based on the rounding-off result. Algorithm 2 depicts the process of adding the extra convolution, where n is the number of layers, O is the set of all $\{O_i\}$ operations in the encoder, P is the set of pooling operations, s is the current shape of the feature maps, k is the input shape, D is the set of decoder operations, m is the current size of D , and e stores all feature map shapes.

To avoid any shape mismatch, RAMOSS keeps track of the shapes of the feature maps during the construction of the encoder (in vector e) using the following rule:

$$e = \left\{ \left\{ \begin{array}{ll} s \mapsto s/2 & O^i \in P \\ s & O^i \notin P \end{array} \mid i \in \{0, 1 \dots n^l\} \right\} \right\}$$

In short, the shape of the feature maps is initialised to be equal to the input shape.

Algorithm 2: Constructing decoder part. Logical brackets are coloured for legibility. **Highlighted** section shows how one of the limitations of MONCAE is resolved, permitting the use in segmentation problems by adding UNET-specific skip-connection in addition to the other per-layer connections discussed in this chapter.

Data: e, n, s, k, O, P

Result: decoder operations

$D \equiv \emptyset$

```

for  $i = n-1$  to 0 do
  if  $O_i \notin P$  then
    |  $D \leftarrow D \cup O_i$ 
  else
    |  $D \leftarrow D \cup \text{upsample}$ 
    |  $D \leftarrow D \cup \text{add}(O_i, D_m)$ 
    |  $s \leftarrow s \times 2$ 
  end
   $A \equiv (s - 2) \times 2 \in e$ 
   $B \equiv s \times 2 \notin e$ 
   $C \equiv (s - 2) \times 2 = k$ 
  if  $(s \times 2 \notin e) \wedge (A \vee (B \wedge C))$  then
    |  $D \leftarrow D \cup \text{conv}((3, 3), \text{valid})$ 
    |  $s \leftarrow s - 2$ 
  end
   $s \leftarrow 1;$ 
end

```

Then the up-sampling layers are used to facilitate the 2x increase of the shape and an extra convolutional layer with valid padding and 3x3 kernel size to do the -2 subtraction from the shape.

Connections for layers with incompatible shapes:

The second exception happens when two layers with incompatible shapes should be added together based on the encoding. Then, all inbound layer connections to the layer currently decoded are taken together with the shape of the current layer as the reference shape s_r . Then, iterating over the *layers for addition* (\mathbb{I}_a) the shape of each of them is adjusted to match the reference layer using pooling operations for decreasing the shape, up-sampling operations for increasing the shape and adjustment convolutional operation in case the exception from Section 4.1.1 occurs.

4.1.2 Benchmarking framework

Pymoo framework (Blank and Deb 2020) is used to control the experiments better and explore the benefits and caveats of different optimisation algorithms. Pym allows us to plug-and-play a range of different optimisation algorithms like NSGA2 (Deb et al. 2002b), NSGA3 (Yuan et al. 2014), Differential evolution (Price 2013) and many more. Moreover, to facilitate the adoption of the approach, the whole codebase and a means to change

the optimisation algorithm easily is provided as well as the input datasets and means to customise the method. In addition, some dataset loaders are included to be used out-of-the-box, including MNIST (Deng 2012), CIFAR10 (Krizhevsky 2009), Oxford Pets, Cityscapes and more. The framework is modular, so adding new objectives or whole objective functions can be easily performed.

Noticeably, although Pymoo is the main supported optimisation library, it can easily be interchanged with any other optimisation framework with minor adjustments to the code.

Researchers are encouraged to use the tool and report the results of their experiments. This new data would populate the benchmarking framework and foster the comparison approach's robustness.

The multi-objective optimisation is based on the Hypervolume indicator metrics (HV or Z-metric, (Guerreiro et al. 2020)) described in Chapter 3. To refresh it, it is defined as:

$$HV(\mathbf{X}, \mathbf{r}) = \Lambda \left(\bigcup_{\mathbf{x}_j \in X} \{f_1(\mathbf{x}_j, f_1^{ref}), \dots, f_m(\mathbf{x}_j, f_m^{ref})\} \right) \quad (4.1)$$

where $f_1 \dots f_m$ are the m specified objective functions, and f_i^{ref} is the reference point in the m -dimensional solutions space for the objective function f_i . Finally, \mathbf{r} is the vector containing all reference points and X is the set of proposed solutions $\mathbf{x}_j \in X$, where each \mathbf{x}_j is an m -dimensional vector characterising the model, and Λ is the Lebesgue measure (Guerreiro et al. 2020).

The defined search space and hyperparameters are easy to adjust in the implementation. In contrast with some previous approaches (Real et al. 2017, Lu et al. 2019, Liu et al. 2019a), the space can be seamlessly expanded to search for other hyperparameters and for using new datasets. In the experiments, a maximum of 12 layers is specified based on the range resulting final layers and the average maximum number of layers used in the selected approach for comparison. See Section 4.3 for further discussions on the hyperparameters employed in each experiment. The size of the original search space of possible parameters for the CIFAR-10 search is 2.4×10^{64} , or 4.4×10^{24} excluding connections. This vast search space suggests the need for a better initialisation technique to facilitate the practical exploration of the optimal space regions.

An alternative approach is to first decrease the search space size to 1.8×10^{28} by ignoring the parameters controlling the number of filters and number of nodes and building architectures with the maximum number of filters. Then, pruning and dropout techniques can be used to remove the unused parameters (Blalock et al. 2020). This idea is out of the scope of this thesis, but it is a promising direction that should be addressed in future work.

4.1.3 Progressive Stratified Sampling (PSS)

Neural architecture search typically requires a vast amount of computational resources (Stanley et al. 2019b). Thus, training NAS approaches, even for a few generations, can result in GPU computational costs in the order of days (Real et al. 2018). For example, consider one of the experiments consisting of computing 10-20 generations with a population size of 20. Each model is trained for five epochs before it is evaluated on the validation set. In this setting, reducing the training time per epoch by just 1 second reduces the search cost by 2000 seconds. Motivated by this result, a new method is devised that is termed Progressive Stratified Sampling (PSS), which allows us to effectively train models on a subset of the data through downsampling the dataset by a customizable factor (5-10 in the experiments). The process, described in Algorithm 3, shows how the dataset splits operate during training and validation.

In short, the models are trained by iteratively picking one of the predefined PSS splits as the generations progress. This method allows models from each i^{th} generation to compete pretty by training on the same data D_{pss}^i split but, at the same time, to be evaluated on unseen data before (the next PSS split D_{pss}^{i+1}). This technique would render a good approximation of the model performance if it were trained with the entire dataset (see results section).

For multi-class problems (such as CIFAR10), the strategy is easily implemented using standard k-fold stratification sampling processes (such as the one provided by Scikit-learn⁶). However, for semantic segmentation or multi-label problems stratification is not defined. Thus, stratification in the context of semantic segmentation is implemented by creating the dataset splits, as described in Algorithm 4, where cl is one of the target classes, and rec is the record being processed.

Algorithm 3: Evaluation process with PSS, where D_{pss} is one of the PSS splits of the dataset, and i is the generation index.

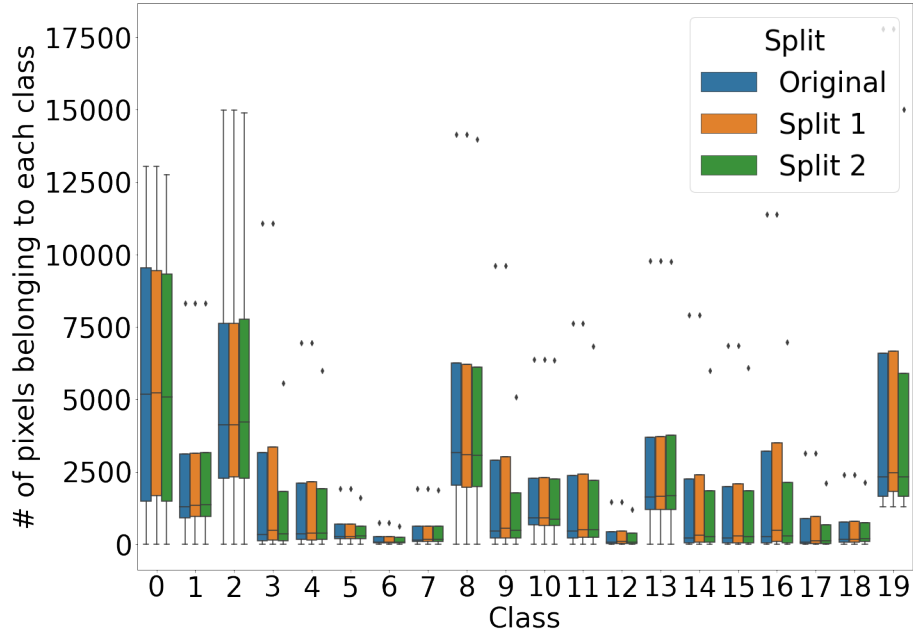
```

Data:  $D_{pss} \leftarrow dataLoader(D_{pss})$ 
Function Evaluate ( $model, D_{pss}$ ):
  for  $i \leftarrow 0$  to  $n_{generations}$  do
     $\widehat{model} \leftarrow train(model, D_{pss}^i);$ 
     $r \leftarrow eval(\widehat{model}, D_{pss}^{(i+1)\%n_{pss}});$ 
    ...;
  end

```

⁶https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Figure 4.3: Mean, standard deviation, maximum and minimum values of pixels of each class compared between two PSS splits and the original dataset. Only two out of the ten splits are shown to enhance legibility.



Algorithm 4: Generating PSS splits (*cl*: a target class, *row*: current image).

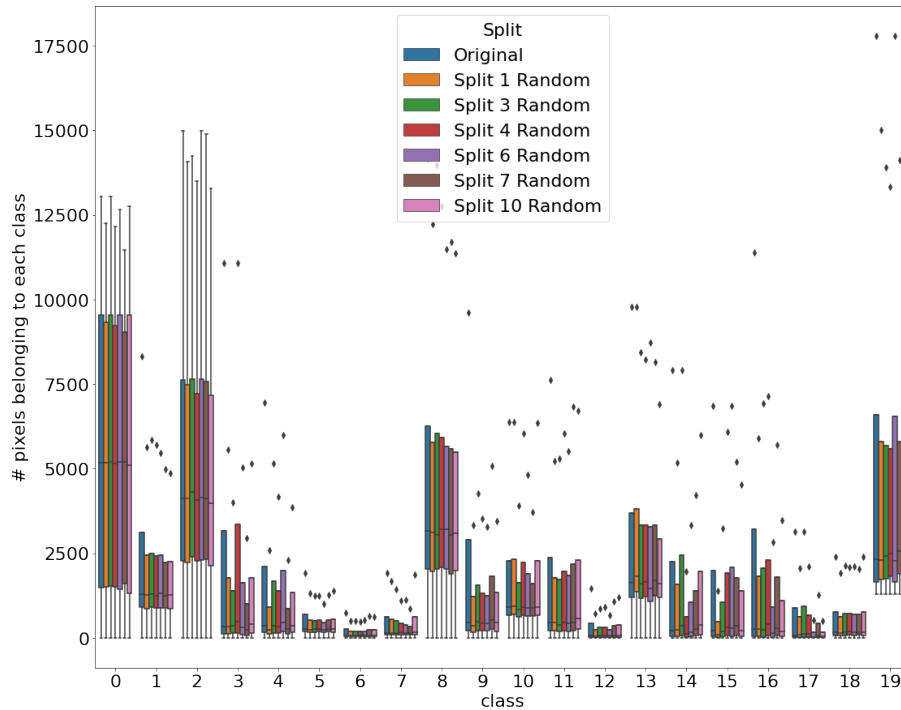
```

Data:  $D \leftarrow data$ 
 $D_{table} \leftarrow loadAsTable(D)$ ;
for  $cl \leftarrow 0$  to  $n_{classes}$  do
  |  $D_{table}[cl] \leftarrow countPixels(D, cl)$ ;
end
while  $length\ of\ D_{table} > 0$  do
  | for  $i \leftarrow 0$  to  $n_{splits}$  do
  | | for  $col \leftarrow 0$  to  $columns\ of\ D_{table}$  do
  | | |  $row \leftarrow max(D_{table}[col])$ ;
  | | |  $split \leftarrow split \cup row$ ;
  | | |  $D_{table} \leftarrow D_{table} \setminus row$ ;
  | | end
  | |  $D_{pss} \leftarrow D_{pss} \cup split$ ;
  | end
end

```

As a representative example of this algorithm, Figure 4.3 shows the boxplot diagram of two of the PSS splits for the Cityscapes dataset compared to the original data. The x-axis (0-19) shows the 20 classes in Cityscapes, and the y-axis indicates the number of labelled pixels for the corresponding class in each split. A small comparative study is also conducted to examine the benefits of using PSS over random sampling (please refer

Figure 4.4: Mean, standard deviation, maximum and minimum values of pixels of each class compared between several random splits and the original dataset. Only 6 out of 10 splits are shown for legibility.



to the next section). As illustrated in Figures 4.4 and 4.3, the distribution of pixels per class in each split is very close to the original one, with the exception of some outliers for a minority of classes. However, although distributions are similar, the key remaining question is how well PSS will integrate with RAMOSS. This will be discussed next in Section 4.3.

4.2 Experimental setup

Before proceeding with the results, it is essential to define the experimental setup, the used dataset and metrics. As mentioned in Chapter 2, one of the most popular metrics for segmentation datasets is the Intersection-Over-Union(IOU) score (Cordts et al. 2015, Chen et al. 2014, Liu et al. 2019a). For classification problems metrics such as accuracy, precision and recall might be adequate to describe the performance, but for problems such as object detection or semantic segmentation, they are not generally used as they fail to capture salient information and successfully inform how well a given model is performing overall and this crucial information is needed for decision making(Goodfellow et al. 2016b). Such problems require more detailed metrics that can account for some margin of error.

The intersection-over-union (IOU) (Nowozin 2014) metric calculates the base overlap

of the predicted bounding box or segmentation mask with respect to the label. This is one of the most popular metrics used in the field of segmentation (Cheng et al. 2021) The first operation is the intersection calculation, and then it is divided by the union, signifying a percentage of overlap. Completing the process over the whole dataset iteratively yields a score for each class. The mean IOU is a single score, which is averaged over the IOU scores of all classes. IOU and mean IOU serve as a more informative measure to understand a model that is predicting roughly where different points of interest are positioned as well as what part of these objects is correctly classified as opposed to simply counting the number of pixels the model got correctly.

The data used here comes from a dataset called Cityscapes (Cordts et al. 2015) (example from the dataset presented in Figure 4.5, which is a semantic segmentation dataset composed of 30 different classes.

The dataset can be used for multiple tasks since the provided labels allow researchers and practitioners to use not only the provided segmentation masks in different resolutions but also in a variety of other formats, which are not the focus of this work. Cityscapes is one of the most widely used tasks for semantic segmentation, and currently, state-of-the-art performances on this dataset reach up to 80% mean IOU (Liu et al. 2019a).

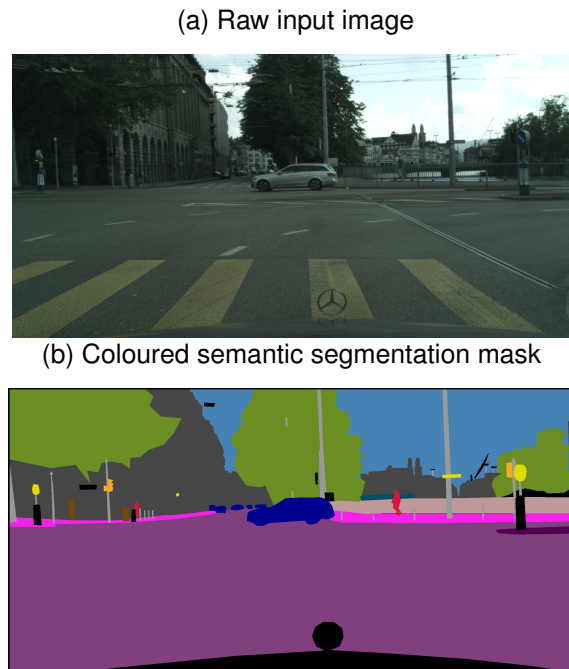
The dataset provides high-resolution images (2048x2048 pixels) mainly from dash cameras, and the photos depict everyday situations on-road or everyday life objects. The data is collected from 50 different cities during different seasons, and all images are in daylight with several weather conditions (Cordts et al. 2015). The dataset also provides valuable metadata, such as GPS coordinates of each image, as well as preceding and trailing frames from the video feed, but the use of the metadata is out of the scope of this study. Labels for pixel-level, instance-level and even panoptic semantic segmentation are provided, and there are 5000 images with fine labels (meaning the pixel-level masks are pixel-perfect or near pixel-perfect) as well as 20000 coarse labelled masks.

The 30 classes in the dataset are split into 8 different categories - flat, human, vehicle, construction, object, nature, sky and miscellaneous (also called void). During the evaluation, not all of the classes are used, and only 20 out of the original 30 are, in fact, responsible for determining how good models are according to the official benchmark (Cordts et al. 2015).

PSS evaluation:

One of the main speed-ups of RAMOSS is achieved through the PSS strategy. Here, the effectiveness of PSS is tested to determine if the training with PSS splits will yield a similar performance ranking of solutions to training with the full dataset. This heuristic will allow RAMOSS to train using PSS instead of the full dataset, which can speed up the method proportional to the chosen number of PSS splits. Stratification is standard in classification problems (Zhang and Wu 2012, Ye et al. 2013). Thus, to thoroughly validate the PSS approach, the focus is shifted to semantic segmentation and, in particular, on

Figure 4.5: A sample image from Cityscapes from the city of Zurich, where a) is representative of the inputs for the model and b) presents what the labels look like.



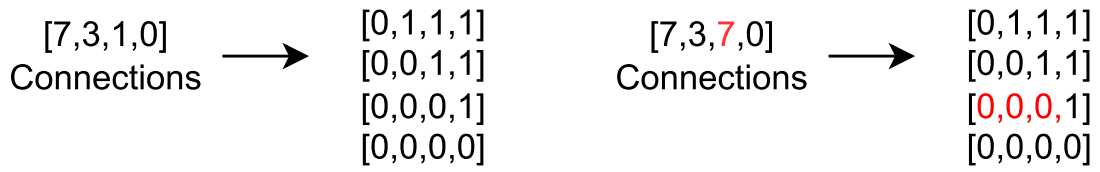
the Cityscapes dataset instead. To this end, 3 common UNet backbone architectures are employed: InceptionResNetv2 (Szegedy et al. 2017b) , InceptionNetv3 (Szegedy et al. 2016b) and EfficientNetb0 (Tan and Le 2019a) operating on the Cityscapes dataset. The networks were first trained using the complete dataset for 25 epochs and then retrained for 25 more epochs.

RAMOSS evaluation:

To test out RAMOSS, first, the search space, the objective functions and some hyperparameters need to be adjusted. Here, as in MONCAE (Chapter 3.4), the same hyperparameters are used following the same intuition discussed in the previous chapter.

The hyperparameters in these experiments are the following: *maximum number of convolutional layers* = 8 (as this results in a maximum of 16 layers including the pooling ones which is a common choice to have increments of 8 blocks like in ResNet-8 (Tang and Lin 2018), ResNet-16 (Duan et al. 2023) and other recent small network success (Gunasekaran 2023)), *maximum number of dense layers* = 3 (as this seems to work quite well in recent studies (Balcioglu et al. 2023, Dhanya et al. 2023)), and both *maximum features* and *maximum nodes* were set to 256 since it is a popular choice among recent (Goldstein et al. 2023, Naseri and Mehrdad 2023, Sriram et al. 2023, Lee and Song 2023) as well as some older foundational works (He and Sun 2015, Dai et al. 2017). The allowed genome properties per each layer that determine the search space are defined in Table 4.1, most of which were motivated by similar works such as Qin and Wang (2019), Lu et al. (2019), Real et al. (2020 2018), Liu et al. (2018) in addition to some preliminary

Figure 4.6: An example of different encoding approaches resulting in duplicate architectures



experimentation done in the process of development that has helped in establishing the computational limitations of the hardware used in the experiments and how close to that limit the experiments can be pushed during the architecture search. Even more importantly, picking these hyperparameters allows for certain constraints to be imposed such that the produced architecture will be comparable to the selected state-of-the-art models. Some of the additional hyperparameters, such as the reference points mentioned in the results tables, were discovered during preliminary experiments with the RAMOSS model during development in an empirical fashion. While similar and possibly better results can be achieved by adjusting these hyperparameters, the ones listed in this work should provide a reasonably good starting point and reliable, reproducible results.

This experiment considered 2 objectives: the categorical cross-entropy loss and the level of complexity, defined as $\mathcal{C} = \frac{\min(\log_{10}(p), 10)}{10}$ in this study, where p represents the number of model parameters. This working complexity measure is inspired by the works of Real et al. (2017) and Dimanov et al. (2021). Thus, this index is based on the number of parameters instead of the number of floating-point operations (like in, e.g., (Liu et al. 2019a)), but future work will explore if both can be used as distinct objectives. Future work should explore the strong intuition behind a potential further improvement of the results by allowing the algorithm to run for more generations or by increasing the population size, but this would naturally result in a higher computational cost.

During the experiments, it was noted that in some cases, different encodings resulted in the same decoded architecture due to the stripping of backward connections (as depicted in Figure 4.6). To address this redundant effect, an alternative was designed to constrain the allowed connections for each layer separately. Specifically, the available values for each layer's connections c_i were redefined as $c_i = \{0, \dots, \max(0, 2^{m_c - l_i} - 2)\}$, where $l_i = \{1, \dots, m_c\}$ is the layer index and m_c is the maximum number of convolutional layers.

Note that for the second to last layer, $l_i = m_c - 1$, as well as the last connections value is 0. This heuristic is intentional because the second to last layer is always connected to the last one since the parameter to disable the model can be used instead of having it enabled and disconnected. This property enabled RAMOSS to drastically reduce the

search space from 2.4×10^{64} to 1.8×10^{28} . In this context, the *maximum number of convolutional layers* was set to 12, the *maximum number of dense layers* to 3, and both *maximum features* and *maximum nodes* were set to 512. The reason for doubling the filters and the nodes from the discussed popular maximum of 256 is that 512 filters is also a popular choice for more complex problems (Marcos et al. 2016, Pons et al. 2017, Juefei-Xu et al. 2017, Masruroh et al. 2023, Sanida et al. 2023). Also, RAMOSS aims to let the architecture better explore a bigger search space since thanks to the sizable reduction this shouldn't hinder the exploitation. Moreover, this showcases how these hyperparameters can be adjusted. The allowed genome properties per each layer that determine the search space are defined in Table 4.1.

4.3 Results and Discussion

Let us now turn to the results for PSS and RAMOSS by first discussing the PSS results.

PSS results:

Figure 4.7 displays the training of the different runs for each of the architectures. From the training performance, it is evident that the training is stable, and they manage to converge. Although, there is a general tendency for lower performance when using PSS, the gap throughout the architectures appears consistent which is the success criteria of the experiment. Next, using the same hyperparameters, they were trained again for 25 epochs, but this time, every five epochs, the training set corresponded to one of the 5 PSS splits of the dataset. Convergence and overall performance are then compared in validation loss for each approach on the same validation set.

To observe the training more closely, the convergence of the models can be seen in Figure 4.7. Remember that the PSS approach trains using just 20% of the data, while control networks train on the full dataset. Thus, the PSS-trained nets' performance is expected to be lower, but this experiment specifically inquires if the architectures' performance ranking stays the same regardless of their precise performance. Consequently, network architecture selection can use PSS splits instead of the whole dataset, drastically reducing the computational cost. Figure 4.8 shows the results for 5 runs with different random seeds, where the loss on a held-out test set is used for evaluating the performance. While the runs with the full dataset take **8 minutes** to complete per architecture per seed, the ones with PSS take only **2.8 minutes**, almost three times faster (note that a 5x speedup is not observed since the full validation set was used for performance evaluation).

More importantly, the PSS approach did not compromise the ranking of architectures based on their performance (Figure 4.8): although there is an expected performance gap between the PSS-based runs and the ones without it, the overall ranking of the architectures based on their mean performance remains. The standard deviation of the

Figure 4.7: Convergence of models during training with and without PSS . Blue are the training runs on the full dataset, whereas the red lines are the runs with PSS.

(a) Accuracy during training with Inceptionresnet2 (b) Accuracy during training with InceptionNetv3 (c) Accuracy during training with EfficientNetb0

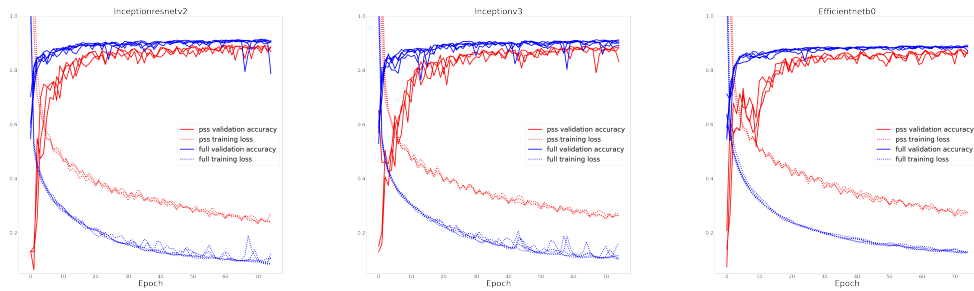
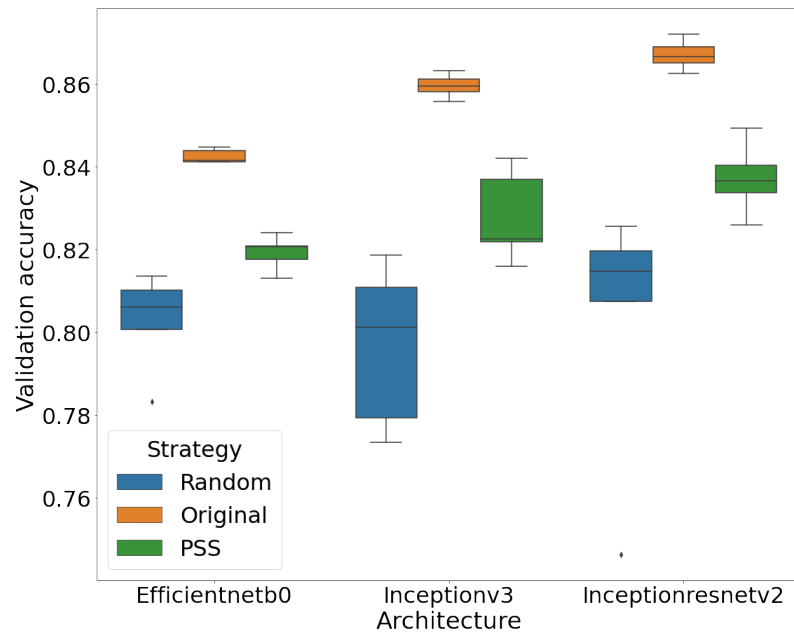


Figure 4.8: Validation loss for PSS-5 and PSS disabled on Cityscapes with UNet with three different backbone architectures.



performance increased with the PSS splits, which is simply due to the fact that models did not manage to converge for this amount of training steps.

To verify the value of the PSS approach, the approach is compared to generated random splits instead of PSS ones. As evident from Figure 4.4, there is a severe discrepancy between the distributions of the number of pixels per class in the different random splits in contrast to the ones from Figure 4.3 where the distributions are substantially closer to the original. Only 6 out of the ten random splits are displayed to enhance legibility. The different architectures are then trained following the same sample size and methodology from Section 3.3. The results are presented in Figure 4.8, where the benefits of PSS can be clearly observed. Looking at the Figure, the ranking of the architectures is not conserved; thus, random splits cannot be used to establish a stable proxy to determine how good a particular architecture is without fully retraining it, which is not the case with the PSS splits.

RAMOSS results:

Next, the performance of the new RAMOSS framework is demonstrated on the image classification problem CIFAR10 (Table 4.2). Interestingly, the results in this table were obtained after only 20 generations with a population size of 20 following the same motivation and intuition discussed in Chapter 3.4 with MONCAE), rendering just 400 model evaluations in **less than 8 GPU hours**.

During the search, an observation was made that RAMOSS gains hypervolume with each new model, suggesting that, in this search space, there are even more optimal solutions than the ones discovered here (Figure 4.9).

Table 4.2: CIFAR10 results. RAMOSS runs for 400 model evaluations (20 generations with a population size of 20) with reference points set to (25, 10, 10) for the evaluation of solutions (Dimanov et al. 2021), see details in the main text. † is used to denote manual architectures; hence, their cHV is undefined "-". The 0.33 GPU days for search operations exclude the retraining of the network for 300 epochs, which took 2 additional Nvidia 3090 RTX GPU hours. RAMOSS achieves an excellent cost-error-parameters balance (**bold font**: best values).

Architecture	Test Error (%)	Parameters (M)	Cost (GPU days)	cHV
	Lower is better	Lower is better	Lower is better	Higher is better
†Wide ResNet Zagoruyko and Komodakis (2016)	4.17	36.5	-	-
ENAS Pham et al. (2018)	4.23	21.3	0.5	35
NAS Zoph and Le (2016)	4.47	7.1	10000+	-
AmoebaNet Real et al. (2017)	5.4	5.4	3150	0
NSGANet + marco Lu et al. (2019)	3.85	3.3	8	83
†Improved ResNet-38 Liu et al. (2020b)	5.83	1.39	-	-
RAMOSS s1	5.49	1.37	0.33	715

It is worth noting that a natural upside of multi-objective optimisation in general, and specifically of this approach, is that it generates a set of solutions instead of just one. This set enables researchers to choose the ideal architecture for their specific needs since this method informs of the trade-offs involved. The set of these approximate solutions gener-

Table 4.3: Cityscapes results. RAMOSS uses 40,50 and 15 as reference points. "*" indicates multiple backbone architectures and "-" for cHV means the solution is fully dominated by another one and does not contribute to the overall hypervolume. Similar to Table 4.2, this table presents the trade-off between the objectives using the cHV (see Table 4.2 for further details).

Architecture	Mean IoU	Search Method	Parameters	Cost	cHV
Auto-DeepLab-L Liu et al. (2019a)	80.3	Gradient-based	44.42M	3	267
DeepLabChen et al. (2014)	63.1	Manual	25M+*	-	-
DeepLabv3 Chen et al. (2017)	81.3	Manual	25M+*	-	-
ResNet-38Wu et al. (2019b)	78.4	Manual	24.46M	-	-
SqueezeNAS LAT XL Shaw et al. (2019)	75.19	Gradient-based	3M	11.5	0
SqueezeNAS MAC XL Shaw et al. (2019)	74.62	Gradient-based	1.8M	14.6	0
MobileNetV3 Large Howard et al. (2019)	72.6	Manual	1.51M	-	-
RAMOSS+PSS s1	76.3	Neuroevolution	1.14M	0.4	8327

ated from the CIFAR-10 runs is presented in Figure 4.9. Looking at the CIFAR-10 results, Table 4.2 shows that, for a fraction of time, the RAMOSS error rate is still competitive with results of highly computationally expensive methods while providing the best trade-off. Crucially, RAMOSS lies on the optimal front of non-dominated solutions (Guerreiro et al. 2020), and thus it discovers an optimal architecture an order of magnitude faster than, for instance, NSGANET. Moreover, the ENAS solution has over ten times the number of parameters of the parsimonious architecture discovered by RAMOSS. The CIFAR-10 model produces 2.7 MFLOPs, which is several orders of magnitude faster than AmoebaNET's 1-1.2 BFLOPs (1-1.2 billion multiplication operations), NSGA's 1.2 BFLOPs and even ENAS's 533 MFLOPs. This optimal trade-off can be quantified by the contributing hypervolume indicator (cHV) (Guerreiro et al. 2020) in Table 4.2. In short, the cHV measures the contribution of each *non-dominated* solution to the hypervolume (see details in, e.g. Guerreiro et al. (2020), Rostami (2014), Dimanov et al. (2021), dominated solutions have 0 cHV). The worst scores from the selected approaches per objective are used as reference points to compute the cHV, like in Chapter 3.

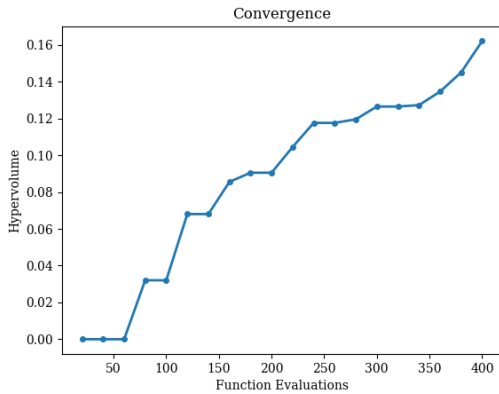
To further test the efficiency of the approach, it is tested on the popular benchmark for NAS algorithms -NASBench-101. To do so, the encoding is adjusted to handle the limits of NASBench. After removing the necessary hyperparameters from the search space so that all architectures from the new search space comply with the constraints of NASBench, Figure 4.10 presents the averaged results over 10 runs for 50 generations with a population size of 20, which equals 1000 function evaluations.

As a reference, the approach is compared to Random Search, which uses the same encoding, but is not subjected to evolutionary optimisation. Both methods achieve a test error rate under 7%, but the NSGA-II error is 5.8% vs 6.2%, the best for Random Search while keeping the number of parameters under 7 million.

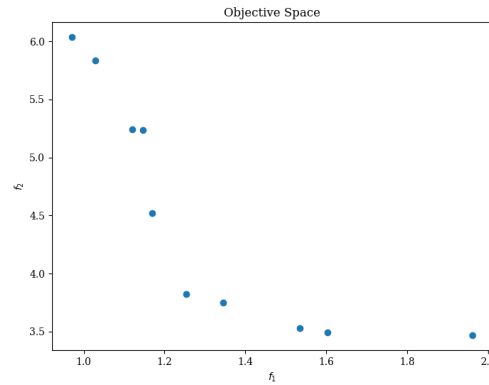
Finally, RAMOSS is ran on Cityscapes for 20 generations with a population size of

Figure 4.9: RAMOSS results on CIFAR10. Figure 4.9a shows the convergence of RAMOSS on CIFAR 10 for 20 generations run with a population size of 20, and Figure 4.9b illustrates the non-dominated solutions in the objective space found during the same run where f_1 is the loss objective during the search and f_2 is the level of complexity

(a) Convergence of RAMOSS on Cifar10



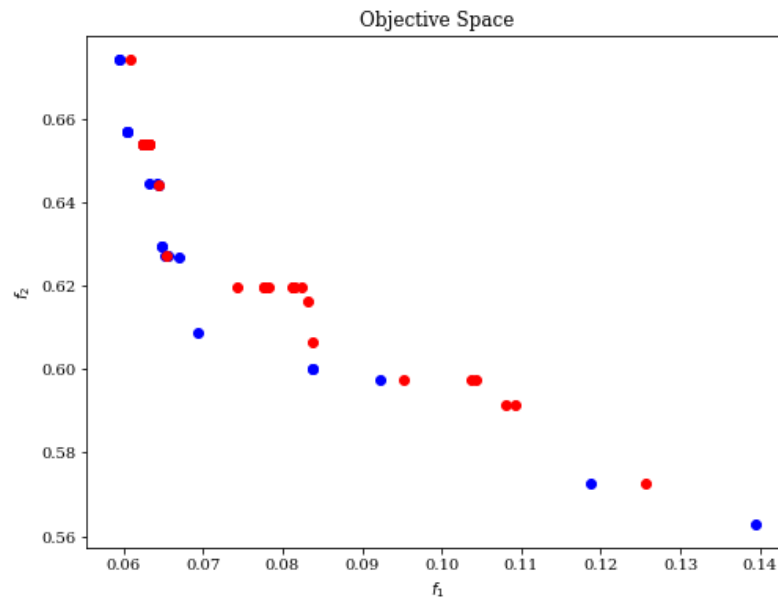
(b) Non-dominated solutions in the objective space



20, while the maximum number of layers is set to 25 (again in an attempt to facilitate better comparisons to the rest of the selected methods), which results in a 76.3% mean Intersection-Over-Union (IoU). Note that, as shown in Table 4.3, this result is not only competitive with the current state-of-the-art nets for this dataset, but it also surpasses some of the expert-designed architectures. Even more strikingly, the optimal architecture discovered by RAMOSS is over an order of magnitude smaller in terms of parameters than all of the reference models (Table 4.3).

The total search time for the selected architecture was 9.5 GTX 1080 GPU hours, with PSS set to 10 splits. The GPU resources were not used optimally during training, and thus the data loading efficiency can be significantly improved to bring the computational cost down even further. Nevertheless, RAMOSS discovered an architecture that is highly competitive with state-of-the-art in a fraction of the time compared to reference approaches. To put how faster the discovered architecture is, and why the number of parameters is an important objective, the number of FLOPs are tested and in line with the number of parameters, the segmentation model that deals with the high-dimensional Cityscapes data turns out to be faster than the CIFAR-10 model by having only 2.3 million multiplication operations or 2.3 MFLOPs. As with the CIFAR-10 model, this is substantially faster than the 80 BFLOPs required by DeepLabv3's model and the 90-110 MFLOPs of the similarly sized models. It is worth mentioning that the smallest SqueezeNAS has similar amount of FLOPs, but it's meanIoU is 68%. Altogether, results suggest that this algorithm displays the ability to effectively conduct multi-objective optimisation by discovering optimally parsimonious architectures, drastically reducing the computational cost.

Figure 4.10: NASBench Pareto-front of found solutions. Since both f_1 and f_2 are minimisation functions, notice that RAMOSS with NSGA-II(Blue) outperforms random search(Red) by presenting a front of solution with a higher hypervolume.



4.4 Conclusion and Future work

This approach is devised to be a stepping stone toward the transition from manually designed neural architectures to automated machine learning. RAMOSS achieved state-of-the-art results while balancing computational cost and test error more effectively than existing semantic segmentation methods. These results showcase the importance of multi-objective optimisation for AutoML. Expanding the open-source benchmark with other datasets could facilitate research in NAS and AutoML areas, where the entry barrier for practitioners is relatively high.

The results show that, despite the need for further improvements, RAMOSS competes with the state-of-the-art and even outperforms it in some cases, offering a good starting point for model selection when presented with new datasets and problems. More importantly, it provides an optimal trade-off between performance and computational resources with respect to the approaches analysed. However, for the NASBench case study, random search is used as the baseline. Future work should compare RAMOSS to a broader range of alternatives (such as, e.g., Real et al. (2018), Elsken et al. (2018a)) and with reinforcement learning strategies. It is also essential to study further the role of two primary hyperparameters - the population size and number of generations; as well as the effect of different optimisers.

To conclude, the framework presented here is a contribution to more effective cross-algorithm, cross-encoding comparison approaches for NAS. More broadly, it can render a general and efficient multi-objective methodology for NAS approaches in computer vision

problems. The next chapter will explore how the process can be further optimised and scaled, so it becomes compatible with real-world data and problems.

Chapter 5

Applications of RAMOSS and MONCAE to Concealed Weapon Detection

The two previous chapters discussed new effective neuroevolution methods designed for addressing some quintessential computer vision problems from a holistic perspective. This chapter, by contrast, focuses on how they can be applied in the context of real-world practical security applications.

The chapter is semantically split into two sections, both of which aim to utilise one of the most extensive concealed threat detection datasets available- SIXray (Miao et al. 2019) to examine and evaluate MONCAE (presented in Chapter 3) and RAMOSS (presented in Chapter 3).

The first one, Section 5.1, showcases the application of RAMOSS to two different concealed threat detection problems via a new automatic evolutionary multi-objective approach specifically designed for concealed weapon detection (MEOW). Through the use of some heuristics (explained in Section 5.1.2), MEOW manages to outperform the current state-of-the-art for both datasets with smaller (ex. 11 million parameters for MEOW vs 23 million parameters for ResNet50) and hence faster (ex. 8x speed-up of MEOW). Moreover, despite this substantially more parsimonious description, it renders better-performing models overall in some cases (e.g., 85% vs 77% mAP for SIXray).

Since the two presented datasets are complex classification tasks and not segmentation-based, a supplementary task is used that is defined on one of them to showcase the performance of MEOW on a real-world threat identification segmentation task. On this task, MEOW achieves the third-best IOU score of 57.7% (in comparison to 59% and 61% for the second and first place, respectively) and an f1 score of 40.58% (in contrast to 42.8% and 43.98% for the second and first place respectively) but using only 530k parameters (compared to 62 and 30 million for the second and first place respectively). Interestingly,

MEOW is the only model to display similar performance during validation and testing, suggesting that it might be a more robust model than the others.

These results (discussed in more detail in Section 5.1.4) represent a significant improvement over previous approaches and further align with the goal of this thesis to make AutoML fast enough for practical deployments, such as in the context of concealed threat identification. The findings also suggest that MEOW successfully demonstrates the feasibility as well as the value of real-world applications of AutoML (discussed in Section 5.1.5).

Until this point in the thesis, the main focus has been on the models rather than the data. However, both problems are not necessarily disconnected (Goodfellow et al. 2016b). For instance, through some relatively minor tweaks in the problem formulation, RAMOSS and MONCAE can be turned into critical dataset discovery algorithms. This is the rationale followed in Section 5.2, which presents a new multi-objective, neuroevolutionary approach for discovering the fundamental set of images in a computer vision dataset (the "coreset" (Dubey et al. 2018)) that is termed MIRA-ME.

Both MEOW and MIRA-ME utilise one of the largest concealed threat detection datasets available- SIXray (Miao et al. 2019). It is a relatively new publicly available security threat dataset comprising over a million X-ray baggage scans. The data is split into 6 classes (Gun, Knife, Wrench, Pliers, Scissors and negative). The dataset was collected from several subway stations (Miao et al. 2019). The dataset is described by the authors (Miao et al. 2019) to have the following 4 important properties: 1. Presence of overlapping objects 2. Inherent intra-class variation from the different shapes, scales, viewpoints and styles 3. Noise from the heavily cluttered objects, and 4. Heavy class imbalance.

SIXray consists of 3 separate datasets (some extra information about the dataset is presented in Figure 5.1): "SIXray 10", "SIXray 100" and "SIXray 1000", denoting 1:10, 1:100 and 1:1000 ratio of threats to benign images, respectively. This categorisation means that in "SIXray 1000", for every image of a prohibited item, there are 1000 benign ones. Such a huge class imbalance can easily overwhelm a classifier, and the class imbalance has to be considered when designing a proper approach. More specifically, when choosing the right loss and metrics.

In short, in the author's view, the results of this chapter suggest not only the "plasticity" of the techniques proposed in this thesis to provide industrial impact but also attest to their capacity to address some caveats of the state-of-the-art in the "model space" (MEOW) and in the "data space" (MIRA-ME, section 5.2.1). Focusing on the latter, two variants (supervised and unsupervised) of MIRA-ME are discussed in detail in Section 5.2.2, which outperformed the state-of-the-art on coreset discovery methods by achieving 97.67% and 65% accuracy on MNIST and CIFAR-10 respectively, and 62.03% mAP on SIXRAY with 3536 images (compared to the second best of 46.68% mAP achieved by Glistler with 14992 images) (see section 5.2.4).

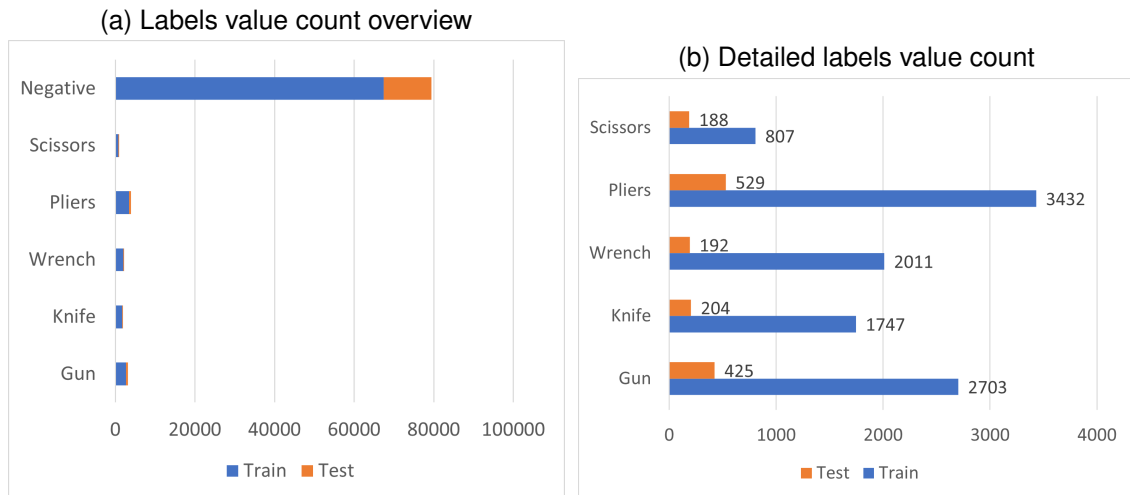


Figure 5.1: Sixray class labels. Notice the severe imbalance between the "Negative" class and the rest. Also, the "Scissors" class is the most undersampled class, and the training and testing samples seem to follow a similar distribution.

5.1 MEOW - Automatic Evolutionary Multi-Objective Concealed Weapon Detection

Concealed weapon detection through screening procedures is a crucial component of public security, especially in crowded areas Mahajan and Padha (2018), Xue and Blum (2003). The need for such systems has significantly increased in the last couple of decades due to, e.g., the rise in terrorism threats globally Sheen et al. (2001), as well as the rising number of school shootings Freilich et al. (2022) and weapons trafficking (Langlois et al. 2022) in some countries.

Many new systems have emerged, focusing on a wide range of different techniques to overcome this uprising issue (e.g. Rostami et al. (2015), Agarwal et al. (2015)). The fact that regulations and techniques to address this problem date back more than two decades connotes the complexity and arduousness of practical solutions for detecting concealed threats (Bartley and Cohen 1998). Most automated state-of-art algorithms in this domain focus on identifying if there is a threat or not, which is a sub-optimal approach since different threats require different security protocols (Morris et al. 2018, Petrozziello and Jordanov 2019).

More recently, some new algorithms are starting to address this problem by combining the effectiveness of deep neural networks with evolutionary computation, which may lead to a breakthrough allowing a system to successfully identify different threats with sufficiently high accuracy (Rostami et al. 2015). Usually, millimetre wave detectors and metal detectors, etc. (Rostami 2014) are used to carry out concealed weapon detection.

There are; however, specific use cases where computer vision algorithms have been employed to analyse raw signals provided by these detection approaches (Mery et al.

2015). One medium of interest in this work is the one which comes from X-ray or CT scans. Usually, these techniques work in tandem with an image processing pipeline, which analyses the scans and determines if there is a threat, sometimes what the threat is and even where it is (Miao et al. 2019). These generated images significantly differ from the visual RGB images generated from CCTV or other digital cameras, as discussed in Chapter 2.5. Hence, they require specific preprocessing to be applied before they can be used as input to computer vision algorithms (Mery and Arteta 2017).

Moreover, conventional image processing and naive application of already established computer vision algorithms often yield sub-optimal results for threat detection. Thus, previous studies agree on the compelling need for domain-specific architectures and procedures (Miao et al. 2019, Mery et al. 2015).

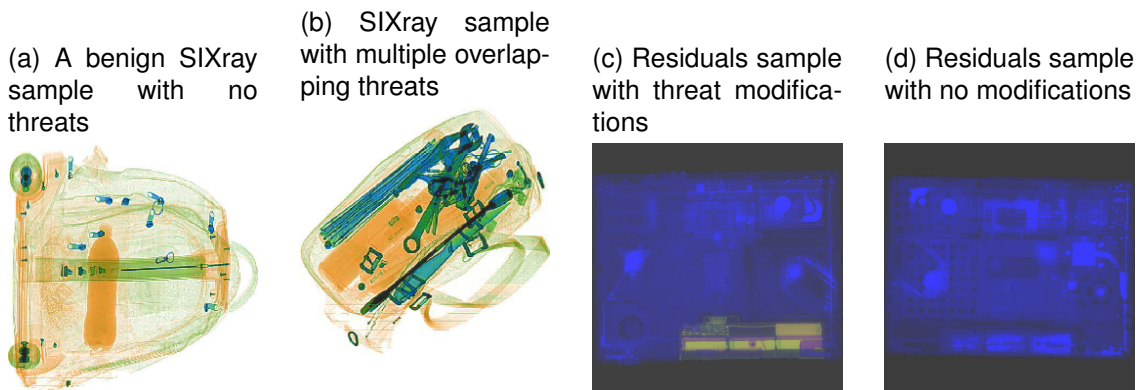
Developing new architectures is typically a complex process, often requiring adaptive tuning to make them fit a specific problem involving extra computational and human resources. This motivated the development of recent techniques invariant to changes in data distributions and other external settings He et al. (2021), Karmaker et al. (2021). The idea of such approaches is to effectively harness the tremendous computational power available nowadays to support and even partially replace a multidisciplinary team of domain experts tackling the problem manually. Automated machine learning (AutoML) allows all of this to be possible, and recently, the progress in the field has led to some remarkable breakthroughs, e.g., Real et al. (2017), Lu et al. (2019), Zhou et al. (2020). One of the significant caveats of AutoML, however, is its extreme computational requirements Lu et al. (2019), Real et al. (2017)

To alleviate these drawbacks, a potential approach to deal with the high computational demand of these algorithms is to incorporate optimisation heuristics in the form of proxy scores, which promise a significant reduction in resource requirements Mellor et al. (2021), Abdelfattah et al. (2021).

Here, the potential application of AutoML approaches for concealed weapon detection is explored using two X-ray datasets that feature different threats and scenarios (samples from these datasets are presented in Figure 5.2). The contributions of MEOW are the following:

1. A novel application of state-of-the-art methods to industry problems in concealed weapon detection.
2. A multi-objective optimisation of several heuristic proxy scores simultaneously, with an up to a 200x speed up w.r.t. standard multi-objective AutoML algorithms.
3. A novel ensemble approach, utilising individual predictions from several optimal models discovered by a multi-objective NAS to provide a final, more accurate prediction.

Figure 5.2: Sample images from the two datasets



5.1.1 Related work

As the interest in the field has grown, more and more research has attempted to fill this research gap and provide alternative solutions to deal with the problem, such as Dimanov et al. (2021), Zhou et al. (2020), Assunção et al. (2019). In particular, RAMOSS (presented in Chapter 4 allows easy integration with datasets such as SIXRay.

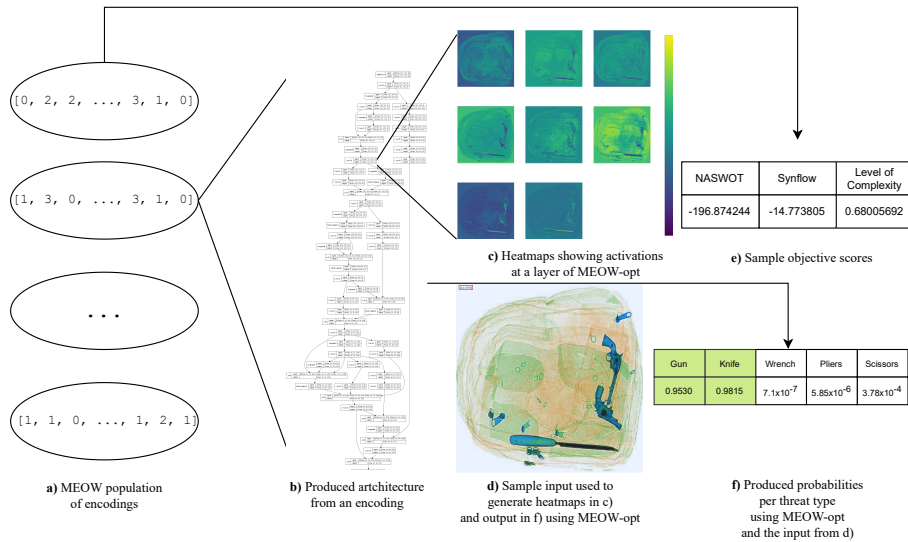
However, RAMOSS, like most NAS algorithms White et al. (2023), spends most of the computational time on training each constructed architecture for a certain amount of epochs to evaluate its performance on a held-out validation set and compare it to the rest of the produced architectures, which is sub-optimal. Real et al. (2017)

Thankfully, recent works have provided valuable shortcuts to alleviate this computational cost by allowing performance approximation proxy scores to be integrated into these algorithms, such as Neural Architecture Search Without Training (NASWOT) Mellor et al. (2021) and Iterative Synaptic Flow Pruning (SYNFLOW) scores Tanaka et al. (2020) introduced in Chapter 2. Even though they do not display a perfect correlation with the trained performance, they provide an accurate estimate of the model's performance Abdelfattah et al. (2021). Although SYNFLOW was initially designed to be a score used for pruning architectures, a recent study Abdelfattah et al. (2021) suggested that it might be an effective performance proxy score when used in unison with NAS approaches. In this chapter, the RAMOSS algorithm is leveraged with these newly surfaced proxy scores to establish the feasibility of using NAS approaches in concealed weapon detection and test how well such approaches scale to real-world use cases.

5.1.2 MEOW Methodology

Here, the recent RAMOSS algorithm is upgraded with newly surfaced proxy scores Abdelfattah et al. (2021), Mellor et al. (2021), Tanaka et al. (2020) to foster its feasibility for concealed weapon detection and test how well such approaches scale to real-world use cases.

Figure 5.3: The MEO algorithm. The population of architectures are encoded using lists of integers (a), and then they are decoded to architectures using RAMOSS (b). Panel b) shows a condensed version of the MEO-s1 architecture for SIXray. Sub-figure c) presents the activations of a selected layer of the trained MEO-s1 when a sample input (d) is passed through the network. e) presents a sample objective scores of an architecture in the population, which is used for selection. Finally, f) shows the raw predictions of MEO-s1 for input d).



In addition to the SIXray (multi-label) dataset, the pipeline is run on a proprietary dataset containing different types of threats and modifications. In essence, this new dataset was generated using an award-winning proprietary project, which uses a multi-staged computer vision algorithm to match and identify anomalous devices and then classify these anomalies together with the types of modifications in devices.

Next, using the Residuals dataset, the chapter explores how the current state-of-the-art performs in multi-output scenarios where the model is requested not only to predict the type of modification to a particular device but also to identify what type of threat (if any) is present. Even though the Residuals dataset shares many similarities with SIXray and both datasets make use of X-ray technology to generate the images, with the Residuals one, a residual X-ray image of the discovered anomalies is utilised as input as opposed to the raw pseudo-coloured X-ray scan as is the case with SIXray. However, a challenge in using NAS approaches like RAMOSS for threat detection is their computational cost, e.g., SIXray alone has over a million data points.

Thus, motivated by the promising results of "Zero-cost NAS" Abdelfattah et al. (2021), a rapid multi-objective neuroevolution approach capable of finding optimal architectures in less than an hour of computational time called MEO is devised (a general representation of process is available at Figure 5.3). More specifically, the approach uses two of the proxy scores presented in Abdelfattah et al. (2021) - NASWOT Mellor et al. (2021) (labelled *jacob_conv* in Abdelfattah et al. (2021)) and SYNFLOW Tanaka et al. (2020). In

contrast to "Zero-cost NAS", MEOW does not use a cumulative score to aggregate over the selected proxies. Instead, MEOW uses them as separate objectives and optimises for both simultaneously.

In short, the NASWOT and SYNFLOW scores are reformulated from a pruning metric to a performance estimation proxy for assessing a specific architecture, like in Abdelfattah et al. (2021). The overall NASWOT score (NS) per architecture is thus estimated as

$$NS = \ln(|\det(K)|), \quad (5.1)$$

where the symmetric matrix K of dimensions $n \times n$ (n = number of images for the current input mini-batch \mathbf{X}) represents the similarity between inputs for a given l -layered architecture to be assessed (see details in Mellor et al. (2021)). Its entries $K_{i,j}$ are binary values characterising each pair of input images i, j . The following score calculation only considers layers of rectified units by the l^{th} -layer activation function f_l , and hence the output (softmax-activated) layer is excluded. The underlying rationale of the approach is that the more dissimilar the binary codes associated with each pair of inputs i, j are, the easier it is for the network output layer to discern between them (Mellor et al. (2021)). Such binary codes are computed as:

$$K = \sum_{l=0}^L \left((g_l(\mathcal{X}) \cdot g_l(\mathcal{X})^T) + ((\mathcal{I}_l - g_l(\mathcal{X})) \cdot (\mathcal{I}_l - g_l(\mathcal{X}))^T) \right) \quad (5.2)$$

where \mathcal{X} is the input to this layer ($\mathcal{X} \equiv \mathbf{X}$ for $l = 1$). g_l is a binary mapping $g_l : \mathcal{R}^+ \cup \{0\} \rightarrow \mathcal{Z}_2$ operating over each single output of the layer l such that,

$$g_l(x \in \mathcal{X}) := \begin{cases} 1 & \text{if } f_l(x) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (5.3)$$

For simplicity, the full tensor containing mappings for layer l outputs is denoted as $g_l(\mathcal{X})$. This tensor is flattened such that $\hat{K}_l := g_l(\mathcal{X}) \cdot g_l(\mathcal{X})^T$ is a $n \times n$ matrix. Likewise, \mathcal{I}_l is a tensor of ones of the same dimensionality as $g_l(\mathcal{X})$. Intuitively, equation equation 5.2 computes the similarity between each pair of images encoded in binary code $\{0, 1\}$ aggregated over layers $1, \dots, N$. The final score summarises this similarity for the entire architecture by computing the (log-) determinant of the matrix K (equation equation 5.1).

The next proxy score considered is SYNFLOW, a *data-agnostic* (training-independent) index which focuses on weights values (instead of the rectified units outputs like NASWOT), and hence it provides a complementary view to estimate the architecture performance. Like in NASWOT, the final softmax layer $l = N$ is also not considered in the SYNFLOW computation.

In brief, the network is maximally stimulated with a responsive input (a "white" im-

age -tensor of ones). In this setting, a low dependency of the network output (at layer $l = N - 1$) on weight magnitudes acts as a proxy for a high discrimination capability of the architecture Abdelfattah et al. (2021). Intuitively, this proxy is indicative of the importance of parameters (weights) in determining the network output, such that sparseness in relevant connections leads to more selective paths of information processing fed to the classification layer N (see details in Tanaka et al. (2020)).

The score $s_l(\theta)$ per-connection (that is, per-weight parameter) θ at layer $l < N$ is termed synaptic saliency Tanaka et al. (2020), which is simultaneously computed for all layer connections $\theta \in \Theta_l$ as

$$S_l(\Theta) = \frac{\partial \mathcal{L}}{\partial \Theta} \odot \Theta \quad (5.4)$$

where \mathcal{L} is the so-called *pseudo-loss* function, simply consisting of the aggregated activation for all units in layer $N - 1$ Tanaka et al. (2020), \odot is the Hadamard (element-wise) product, and $S_l(\Theta)$ is a tensor of synaptic saliencies for all layer connections (note that the derivative in equation 5.4 is computed over each input weight θ of the layer, and it is represented in this compact fashion for simplicity in the description).

Next, following Abdelfattah et al. (2021), all individual scores per neuron i in layer l are aggregated, where the subset $\Theta_{i,l} \subset \Theta_l$ contains the neuron's input weights, to obtain $S_{i,l} = \sum_{\theta \in \Theta_{i,l}} s_l(\theta)$. To conclude, the process is taken one step further by averaging these summarised scores per layer and neuron, $S_{i,l}$, into a scalar index characterising the entire architecture as:

$$\ln \left(\frac{\sum_{l=1}^{N-1} \left(\frac{1}{|\Theta_{i,l}|} \cdot \sum_{i=1}^{|\Theta_{i,l}|} \ln(S_{i,l} + 1) \right)}{N - 1} \right) \quad (5.5)$$

Before the main experiments, an ablation study was conducted to explore different backbone architectures and to determine which ones operate optimally for SIXray and why. This can grant us an understanding of what design patterns should be included in the search space for the "ideal" architecture. The study will also establish baseline models to compare with MEOW models.

The findings from this ablation study would not only serve to compare the performance of different architectures/layer blocks but also provide insights into additional objectives that can be considered when conducting the multi-objective optimisation for the problem at hand. More broadly, this can open up the possibility of establishing methodological guidelines for discovering such qualitative objectives when the approach is applied to real-world scenarios, such as the ones shown next.

5.1.3 Experimental setup

To conduct this study, an Adam optimiser is used with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 1 \times 10^{-8}$ and a conventional categorical cross-entropy loss. The motivation behind these hyperparameters is based on the same reasoning as in Chapter 3. To adjust the architectures to the given problems, the top layers are discarded for all architectures, and the same five dense layers responsible for classifying the produced features from the competing architectures are added to ensure a fair comparison. Categorical cross-entropy is used as loss and also shuffle and fetch the dataset using the same random seeds. Each architecture was trained for 50 epochs and evaluated on the same held-out test set. No augmentation is used during the study to keep as many control variables as possible.

After setting up the corresponding data loaders and computing the benchmarks, the general process can be described by the following steps: **1.** Specify hyperparameters, which include reference points for each of the objectives. The reference points for SIXray and the Residual datasets are empirically chosen to be 600 and 80 (for NASWOT and the SYNFLOW score, respectively) **2.** MEOw algorithm runs for g generations with a population size p . The specified maximum number of convolutional layers to be used in all experiments in this work is set to 35. **3.** Pareto front of produced solutions is taken and the architectures are trained for 50 epochs following the same process as with the other state-of-the-art methods. **4.** The predictions of the models are taken and an ensemble model is constructed that uses the probabilities of all discovered architectures to take a final decision. **5.** All architectures and the ensemble model are evaluated on the dataset at hand. The results report the best-performing architecture score from the MEOw generation (labelled "MEOw-opt" in Tables 5.1 and 5.2) as well as the score from the ensemble models (labelled "MEOw-ens" in Tables 5.1 and 5.2).

The reason for the increase of maximum number of layers allowed when compared to RAMOSS in Chapter 4 and MONCAE in Chapter 3 is mainly based on the dataset and on the state-of-the-art models (Szegedy et al. 2015b, Tan et al. 2019, He et al. 2015). A few runs with running the algorithm with more than 35 allowed layers were attempted but unfortunately some hardware limitations were encountered with the available resources for this project, thus the maximum feasible 35 was finally chosen.

5.1.4 Results and Discussion

The results are reported based on the average precision in the case of SIXray and the two separate accuracies for the two outputs for the Residuals dataset. The reason for using precision for SIXray is not only to match the original SIXray paper Miao et al. (2019) but also because of the massive imbalance in the dataset. With the Residuals dataset, the ratio is relatively balanced. Hence, the section is first going to focus on accuracy and

Table 5.1: SIXray-10 results. Notice that the MEOw-opt and MEOw-ens methods achieve the highest mAP, and the state-of-the-art lags behind significantly. Interestingly, for the most oversampled class, "Pliers" (except "Negative") DenseNet is achieving better performance, but the MEOw architectures balance the rest of the classes more effectively and are especially good at classifying the most underrepresented class "Scissors".

Architecture	AP Gun	AP Knife	AP Wrench	AP Pliers	AP Scissors	mAP
ResNet34	89.71	85.46	62.48	83.50	52.99	74.83
ResNet50	90.64	87.17	64.31	85.78	61.58	77.87
ResNet101	87.65	84.26	69.33	85.29	60.39	77.38
Inception-v3	90.05	83.80	68.11	84.45	58.66	77.01
DenseNet	87.36	87.71	64.15	87.63	59.95	77.36
MEOw-opt	94.93	89.47	67.48	86.13	89.29	85.46
MEOw-ens	95.51	94.04	77.34	76.12	96.34	87.87

then dive deep into the different threat types and the corresponding f1 scores. First, the SIXray results are discussed.

Table 5.1 shows that from the state-of-the-art architectures, the best ones are ResNet50 and ResNet101, which are mostly on par with DenseNet. Interestingly, most architectures display consistent performance throughout all threat classes, although the number of images with the class "Scissors" is the most undersampled one both in terms of training and testing samples (Figure 5.1).

Moreover, DenseNet is the best architecture for detecting Pliers, which is the class with the most samples, excluding the benign ones. However, the best-recognised classes seem to be the Gun and the Knife throughout most architectures.

Further analysis shows a general increase in performance going from ResNet34 to ResNet50, but at the same time, from ResNet50 to ResNet101, the performance drops except for a single class. This phenomenon can be explained by the depth of ResNet101, which may prevent it from converging for the same 50 epochs as ResNet50. Also, it is possible that the model is overparameterised for this particular problem in line with the findings of Pasupa and Sunhem (2016) and Malach and Shalev-Shwartz (2019).

Table 5.2: Residuals results. The residual problem consists of two separate tasks: detecting the type of modification and the threat of the modification. The following table displays the averaged results for both of these separate multi-class outputs and even though marginally, MEOw architectures outperform the state-of-the-art. Remember that the MEOw architectures are substantially smaller than the other ones.

Architecture	Threat Accuracy	Modification Accuracy
Inception-v3	77.52	79.21
Nasnetmobile	68.15	72.96
Inceptionresnetv2	89.06	88.22
Resnet50	88.82	87.26
MEOw-opt	89.18	88.34
MEOw-ens	91.23	89.06

Table 5.3: Residuals results in % f-1 score per class. Here, the detailed per-class scores of Table 5.2 reveal that MEOW architectures, consistently with the SIXray results, achieve the best balance of different classes and are outperformed only for one type of modification by a state-of-the-art model with less than 0.5%. Note that MEOW architectures are substantially smaller than the other ones.

Architecture	Threat					Modification			
	Threat 1	Threat 2	Threat 3	Threat 4	None	Type 1	Type 2	Type 3	Benign
InceptionV3	87.46	87.80	80.36	70.00	71.00	91.13	83.94	60.08	71.30
NasnetMobile	72.22	84.52	40.26	67.06	67.65	86.01	77.20	51.72	66.95
InceptionResNetv2	89.03	89.76	85.71	90.28	89.60	93.54	87.88	78.19	90.06
ResNet50	89.70	89.55	88.70	89.07	87.33	92.49	88.00	77.02	86.96
MEOW-opt	88.74	87.82	87.55	90.40	90.06	93.78	87.66	79.89	88.24
RANISSens	92.11	92.94	91.98	90.27	90.00	94.81	88.02	79.53	89.74

The analysis of the models produced by MEOW indicates that they effectively captured the imbalanced nature of the data, as demonstrated by their performance on under-sampled classes and overall accuracy. This suggests that the multi-objective optimisation process employed by the underlying RAMOSS during architecture search leads to the development of well-discriminative models. This property can be attributed to the influence of the proxy scores, which are designed to reward architectures that can discern distinct or dissimilar features in the data. In this case, it has resulted in models that were particularly effective at differentiating between the various classes. In fact, the best MEOW architecture outperforms state-of-the-art models on the SIXray dataset, even when the class-balanced hierarchical refinement (CHR) technique described in Miao et al. (2019) is used.

The gap between the architecture discovered through the MEOW method and the state-of-the-art calls for further investigation into the importance of designing domain-specific architectures. Further, the MEOW-opt architecture utilises just above **11 million parameters**, significantly fewer than the 23.5 million parameters used by ResNet50. It is possible that the superior performance of the MEOW model can be attributed to its unique architecture, which warrants closer examination of the architecture (See Figure 5.5).

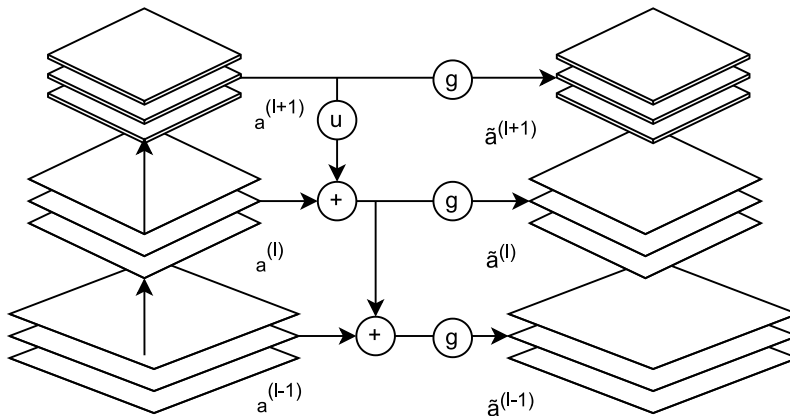
One of the key steps in the CHR Miao et al. (2019) is the hierarchical refinement which utilises low and high-level features by concatenating intermediate activations and then filtering out noisy information based on the signals of the activations of the next layer. Each selected layer for feature extraction thus becomes a separate stream of layer l activations \tilde{a}^l that is fed into an auxiliary classifier $f^l(\tilde{a}_n^l; \xi^l) = y_n^l$ where ξ^l is the hashing vectoriser of the selected layer and finally all y_n^l are averaged to obtain the final output y . Looking back at the architecture discovered by MEOW in Figure 5.5, some large skip connections can be seen. After the first convolutional layer, a skip connection is made to a much lower dimensional representation layer.

Then, the signals are added inside what looks like an auxiliary feature extraction arm,

which is effectively similar to what the CHR accomplishes with the separated streams. Moreover, the architecture seems to be composed of ResNet-like blocks with various degrees of skip connection depth, which, as evident from Table 5.1, seems to be working better than InceptionNet or DenseNet-like blocks.

Interestingly, the models produced from MEOW excelled at recognising different classes, which is attributed to the multi-objective optimisation underneath the algorithm. To utilise a better portion of the generated front of solutions rather than just one (in the form of MEOW-opt), four architectures with the highest contributing hypervolume were selected. Then, a classifier is designed to use the outputs of these architectures $y, y' \dots y'^n$ in order to make a final prediction y_{ens} , which can then be evaluated using the actual labels y^* . The architecture of the ensemble model is custom-designed, and it is composed of a custom layer, the aim of which is to do a weighted average of the predicted probabilities y .

Figure 5.4: Hierarchical refinement process from Miao et al. (2019). a^l signifies the activation a at layer l of the architecture. g is the function from CHR that determines and filters the noise. The $+$ denotes concatenation, and \tilde{a} denotes the filtered activation after g is applied.



The ensemble output naturally achieves the best overall scores, with the sole exception of the most out-of-distribution upsampled class in the testing set - the Pliers, in which it does not outperform DenseNet. Overall, thanks to the ensemble approach, the state-of-the-art was outperformed by more than 10%, which is more than five times better improvement than the one derived from the incorporation of CHR Miao et al. (2019) (a visual representation of the CHR approach may be seen in Figure 5.4).

Next, the state-of-the-art is explored in a real-world industrial proprietary problem and the effectiveness of newly developed AutoML (such as RAMOSS) is tested. First, the accuracy of the two different problems in the Residuals dataset defined in the previous section is examined.

Succinctly, each scan has an associated threat type and a modification type. As this

requires multiple different outputs, all models are fitted with the same top layers (similar to the SIXray experiments). From Table 5.2, it can be observed that the MEOw architectures are once again outperforming all chosen benchmark architectures. Interestingly, the gap between the scores is significantly smaller but still consistent with the SIXray results. Strikingly, judging by the overall results, NasNetMobile and Inception-v3 fail to capture the features of the datasets, which reveals that conventional state-of-the-art architectures should not be treated as a silver bullet to any computer vision problem. Having a 10% drop compared to ResNet is also inconsistent with the results for SIXray. It uncovers a gap in the ability to estimate their performance on a new dataset and the benefits of tailoring an architecture to fit the particular problem.

To investigate the results more closely, the $F1$ scores for each class for both of the outputs are explored as presented in Table 5.3. Unsurprisingly, the MEOw architectures are taking up the top positions. However, in contrast to SIXray, the utility of the ensemble technique here is limited as the best MEOw architecture outperforms the ensemble method for Micro threat modifications as well as for samples with no threat modifications. Even though the results are close and the ensemble method achieves the best overall results, this discrepancy is attributed to lower diversity in the produced approximation set of MEOw for the Residuals dataset. Since only 20 generations with a population size of 20 are used, it is fair to assume that the algorithm did not successfully explore the enormous search space. Future work needs to address this caveat with an ablation study over these two hyperparameters.

It is worth mentioning that using the advancements listed in the previous section (by using the proxy score heuristic), the time needed to run the NAS part of MEOw was reduced to under 1 GPU hour for a population size of 20 for 20 generations, which is first a drastic improvement over the 8 hours required for RAMOSS to discover a segmentation model on Cityscapes with the same hyperparameters (as discussed in Chapter 4) (Dimanov et al. 2022), but also this makes it one of the fastest NAS runs. Cityscapes is also 3 times smaller dataset, making the 8x speed-up even more salient.

Moreover, since there is no training during the discovery phase, it is now possible to run the algorithm without the need of a GPU, since the CPU can handle inference on most machines. This advancement makes the research field more accessible for new practitioners and also feasible to use in a plethora of new domains and industry settings, as suggested by the experimental results.

In this study, the ability of an improved version of RAMOSS (called MEOw) to conduct multi-input/multi-output and multi-label classification problems is demonstrated. However, as discussed in Chapter 4, the original purpose of RAMOSS is to address semantic segmentation problems.

Hence, using a supplementary task provided by the residuals datasets, the problem can be rephrased as a semantic segmentation one using the same multi-input sequence

Table 5.4: Results on Residuals dataset with segmentation masks. In each column the top 3 approaches are highlighted (first, second and third best). While MEOw-opt does not achieve the best test *IOU* and *F1*, it remains the only architecture capable of yielding a validation performance which can resemble its performance on an unseen dataset. Another property making the MEOw architecture potentially the most favourable out of the mix is that it uses only 0.53M parameters, compared to 62 and 30 for the other better-performing alternatives.

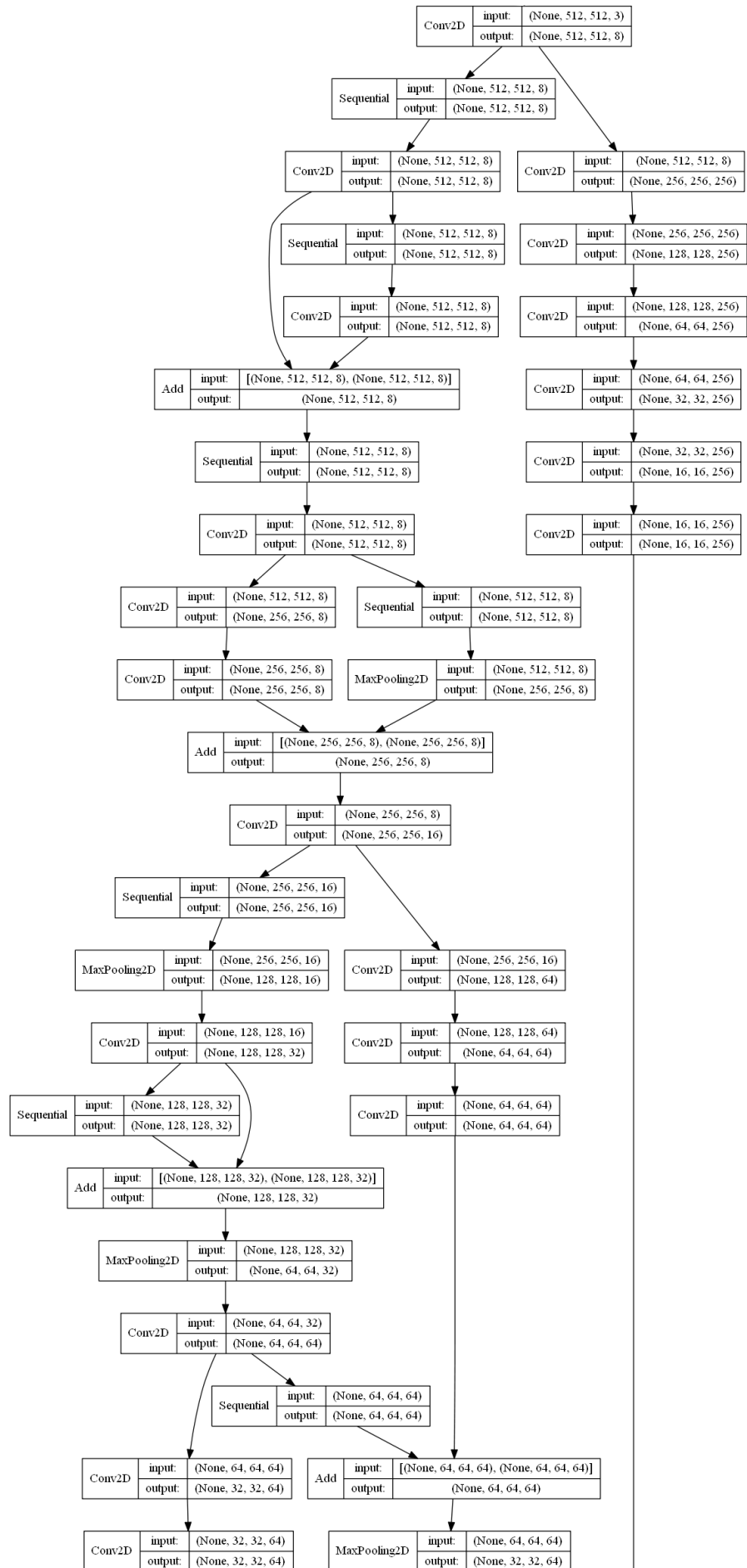
Architecture	Number of parameters (M)	Validation IOU (%)	Validation f1 (%)	Test IOU (%)	Test f1 (%)
MobileNet - UNet	8.34	53.27	69.51	21.60	35.52
VGG16 - UNet	23.75	55.49	71.38	33.48	50.16
ResNet18 - UNet	14.34	54.96	70.93	37.98	55.05
ResNet34 - UNet	24.46	64.46	78.39	37.73	54.79
ResNet50 - UNet	32.56	53.09	69.36	39.18	56.30
InceptionResNetv2- UNet	62.06	62.93	77.25	42.80	59.95
Inceptionv3 - UNet	29.93	59.49	74.60	43.98	61.09
MEOw-opt	0.53	40.04	57.19	40.56	57.72

(i.e., two input images are provided to the network).

The performance of MEOw is then compared with some state-of-the-art methods that are compatible with the dataset, as shown in Table 5.4. Results from the segmentation study are consistent with the ones from Chapter 4. The best-balanced MEOw architecture (MEOw-opt) manages to achieve the third-best results in terms of test IOU and F1 score while being **30-60 times smaller** in terms of the number of parameters compared to the first and second methods. Notably, MEOw-opt is the only one that shows similar results between the validation and the testing sets, indicating that its generalisation error is lower and its training and validation results are more representative of its real-world performance compared to the alternatives. In contrast, the results highlight the potentially detrimental effect of using a well-performing generic architecture on a validation set (e.g., ResNet34-Unet) when it is applied in a real-world scenario. It is possible to speculate that this consistent performance is tied to the fact that MEOw-opt is not over-parameterised like some of the alternatives, although the second least parameterised architecture (MobileNet-UNet) fails to display this ability.

Although the causality of the consistent performance of MEOw has yet to be established, its generalisation capacity is attributed to the fact that the architecture is custom-tailored for the problem at hand. This serves as evidence for the effectiveness of the neuroevolution-controlled optimisation happening in the background, as well as the need for further research on using AutoML techniques for industrial problems.

Figure 5.5: MEOW-opt architecture for SIXray (truncated after the last convolution to enhance legibility).



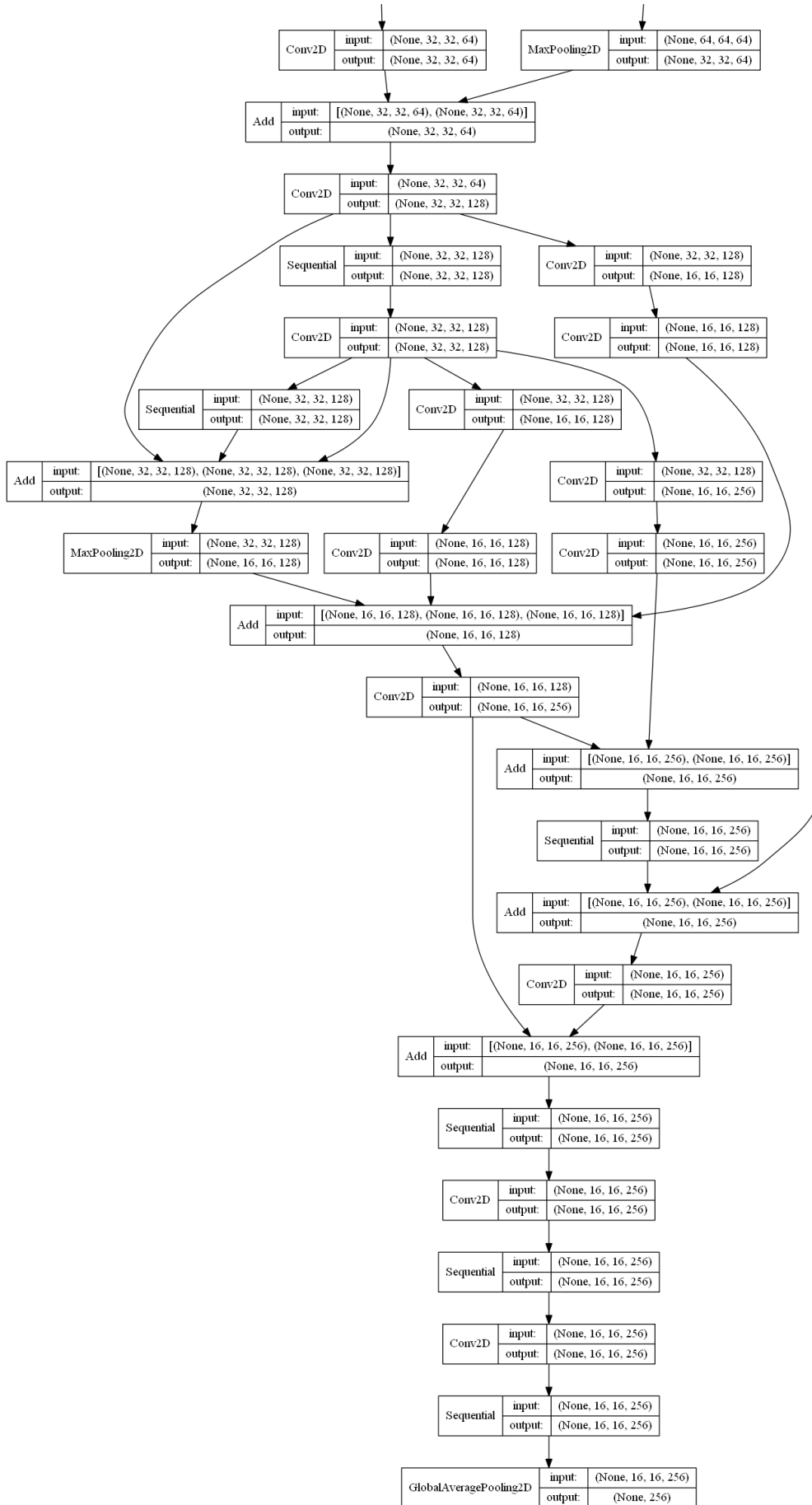
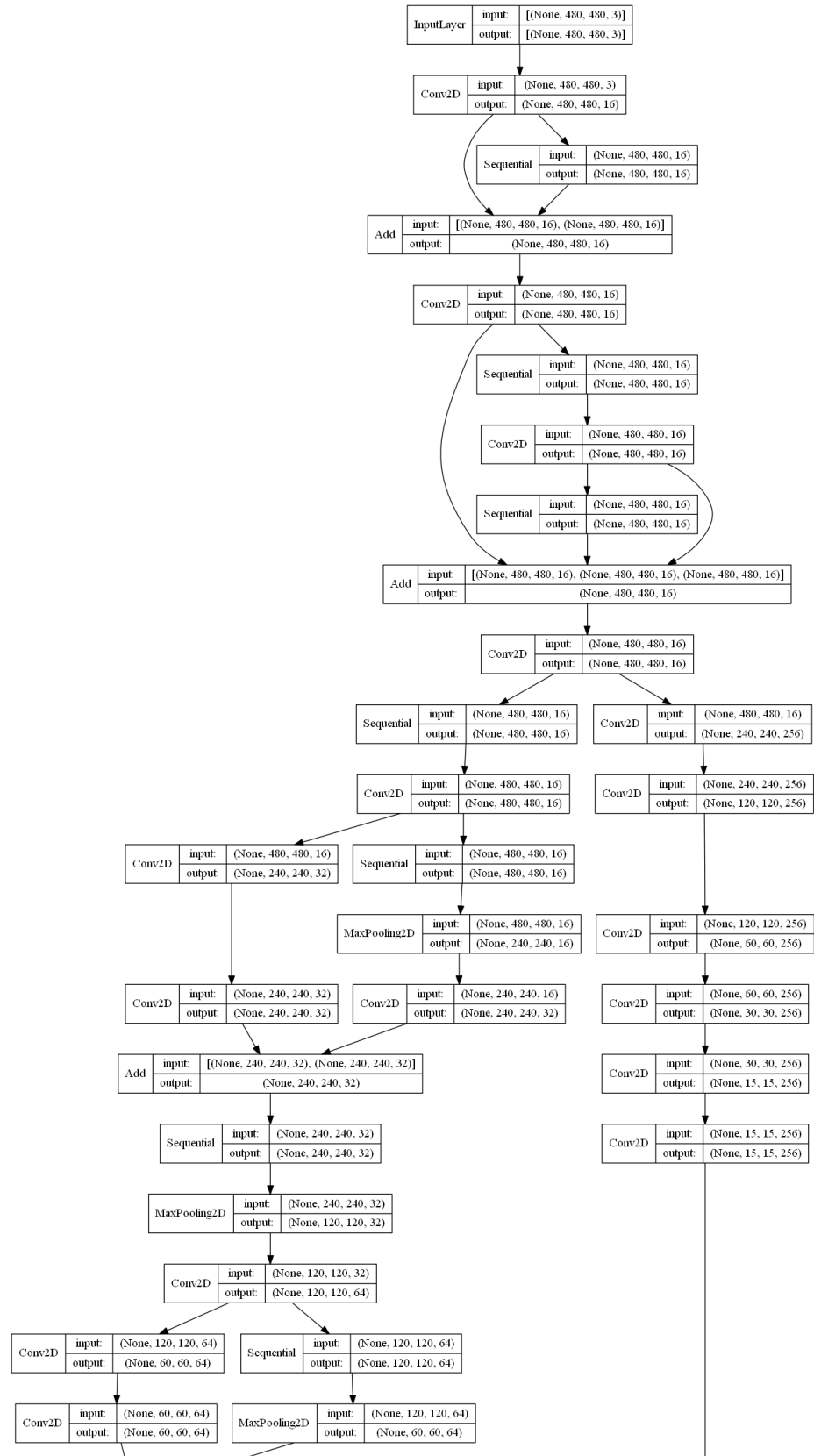
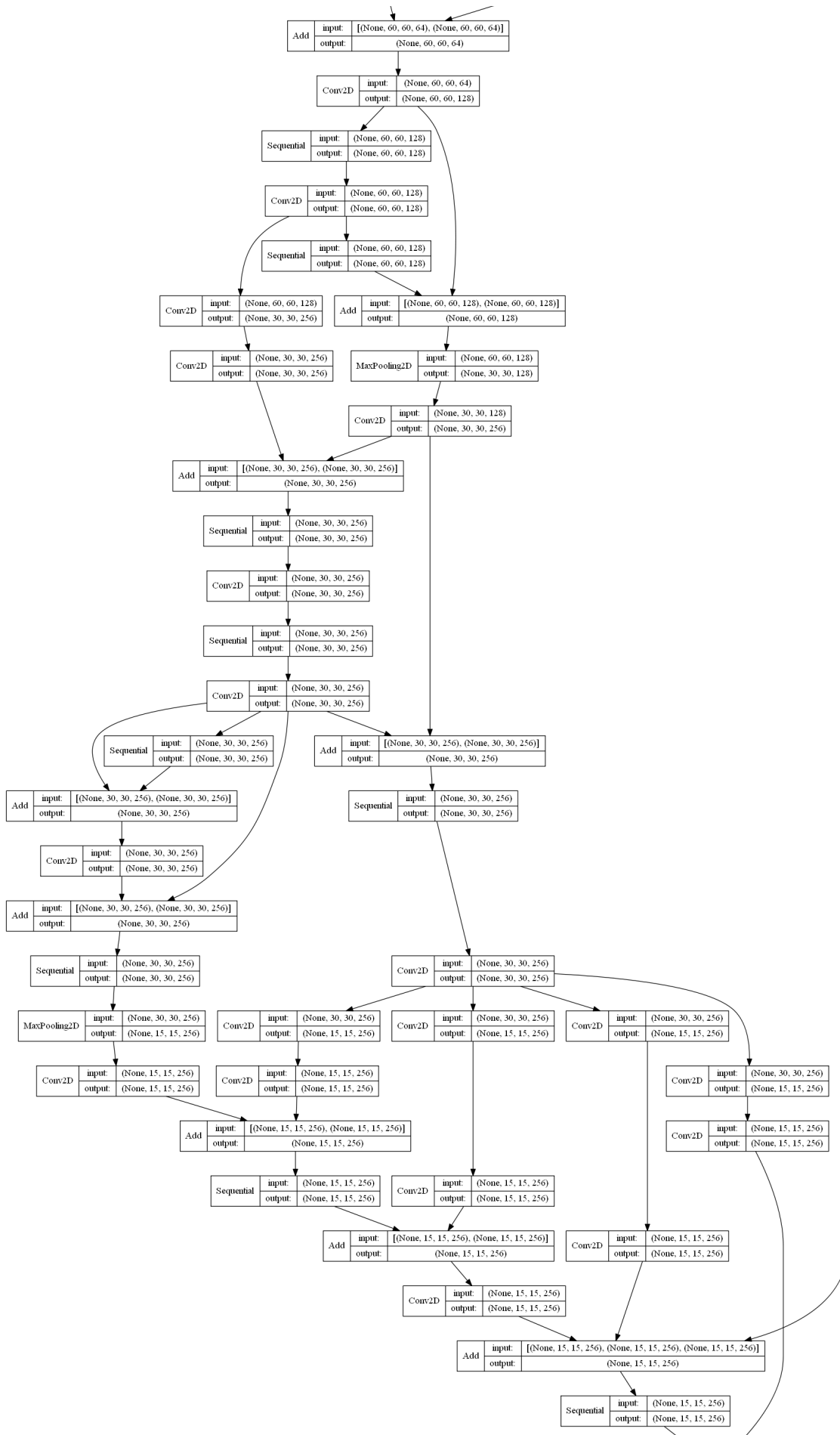
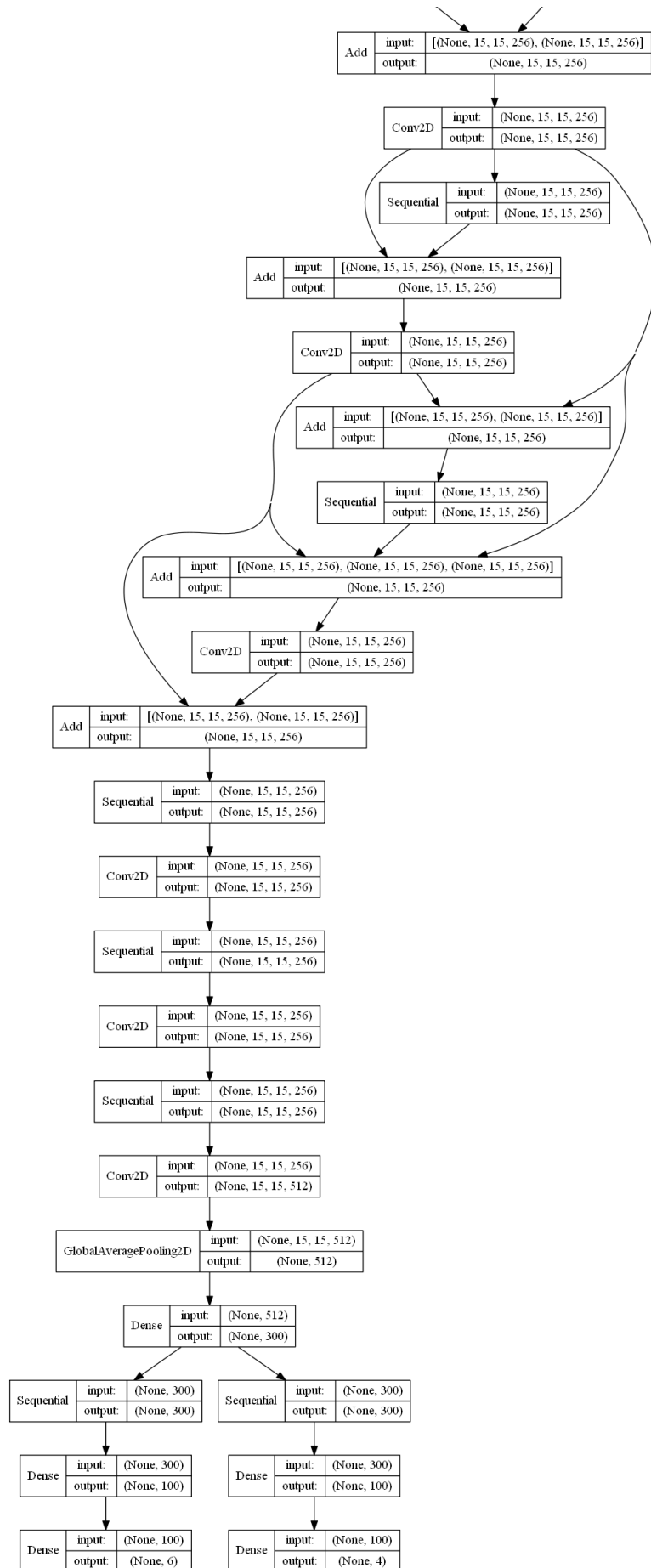


Figure 5.6: MEOW-opt architecture for the Residuals dataset.







5.1.5 Conclusion and Future work

Traditional neural architecture search methods, a particularly fast-evolving, "hot" research topic, have limited use in the industry, given their extremely high computational cost. This study, proposes a new NAS approach to feasibly address real-world problems in threat detection with a substantially lower computational cost. Interestingly, from a practical perspective, the AutoML approach suggested is modularised and hence used in a reasonably straightforward fashion with new datasets, which is not the case with other AutoML techniques (Real et al. 2017, Liu et al. 2018 2019a).

In short, an ensemble approach is designed that leverages multiple sub-optimally discovered architectures instead of disregarding them (as is typically the case). Although the ensemble strategy provides promising results, future work can explore using the collective knowledge of the generated networks to conduct *knowledge distillation* Hinton et al. (2015).

The presented method typically improves the overall performance of the state-of-the-art in both datasets used. Specifically, its behaviour in the most extensive public concealed weapon detection dataset (SIXray) avows the importance of making AutoML scalable to real-world scenarios and designing such systems in a "disentangled" fashion from the dataset used for proof of concept.

Results in both datasets suggest that heuristic performance estimation can drastically improve the computational time of such algorithms, effectively replacing highly resource-greedy evaluation in these settings. A well-performing multi-objective approach is also modified, to use multiple proxy scores to speed up the architecture search and showcase how these proxies can be used in conjunction with multi-objective optimisation to outperform state-of-the-art architectures. In summary, the results and the remarkably low search time (about 1 hour) without the need to use GPUs provide further evidence of the untapped potential of AutoML in industrial applications.

All in all, in this chapter and, more widely in this thesis, multiple datasets and deep learning approaches have been explored. To conclude, albeit obvious, it is worth stressing that understanding how a model operates is entangled with the understanding of the dataset used to train and evaluate it (giving birth to the popular saying in the deep learning community "garbage in, garbage out" (Kilkenny and Robinson 2018, Rose and Fischer 2011, Sanders and Saxe 2017)). Along these lines, the next section in this thesis explores the idea of repurposing this thesis' AutoML efforts to make a contribution to data quality.

5.2 MIRA-ME: Multi-objective coReset discovery through evolutionary algorithms in computer vision

With the new means for generating and collecting data in the last few decades, a plethora of large datasets has recently appeared (Miao et al. 2019, Lin et al. 2014). Some are publicly available (e.g., ImageNet (Deng et al. 2009), SIXray (Miao et al. 2019), and many others Cordts et al. (2015), Lin et al. (2014)) while others are considered valuable intellectual property and are strictly confidential (Gervais 2019).

This influx of data is a great enabler of deep learning Goodfellow et al. (2016b), Surya (2015) since it can potentially foster model accuracy more by considering a larger spectrum of dataset probability distributions. It makes models more robust since the importance of potential outliers during inference can be minimised by incorporating their representative examples in the training set (Limna 2022). Also, the universal approximation theorem generally thrives with more data, which can better describe the underlying distribution Goodfellow et al. (2016b).

However, a new caveat has recently surfaced. Deep learning architecture training is computationally expensive, and each new adjustment typically requires to be tested using the full datasets for optimal reliability Settles (2009), resulting in many GPU hours overall Ju et al. (2022). Contemporary deep learning, especially in the computer vision domain, requires an immense amount of data to perform well, which in turn requires a great deal of human power to gather, label and pre-process this data Ju et al. (2022), Bhalgat et al. (2018).

The process of building and training machine learning models can be time-consuming and resource-intensive and can pose several limitations, including:

1. **Difficulty in rapid prototyping:** Quickly testing and iterating on different model architectures and hyperparameter settings can be challenging, as the training process may be slow and costly (Ju et al. 2022).
2. **High overhead in system upgrades and new projects:** Changing or updating existing models or starting new projects from scratch can be burdensome due to the need to retrain and fine-tune models (Ju et al. 2022).
3. **Overfitting to anomalous patterns** such as a potential class imbalance in the training set not representative of the real-world distribution (Russo and Zou 2019).
4. **Progressively decreasing quality control of the data:** The more data is collected, the harder it becomes to abide by initially set labelling conventions and processes. This discrepancy can lead to inconsistent labelling (like the Ignore label in Kitty) (Zhang et al. 2017b), noisy samples (Valvano et al. 2018) or expensive re-labelling in case of problem reformulation (Goodfellow et al. 2016b) among many

other Valvano et al. (2018), Solorio-Fernández et al. (2020), Ju et al. (2022).

Research dating back to the 60s and 70s Wilson (1972) started looking into different ways to compress these datasets by only using a subsample of the records for training the models while conserving the performance as much as possible. With the explosive growth of data (both in terms of dimensionality and of sheer quantity), new methods started to emerge (Rolnick et al. 2017, Dubey et al. 2018, Killamsetty et al. 2021) and the idea of a "coreset" was coined in Sener and Savarese (2017) . The coreset can be described as the smallest subset of an entire dataset that can be used to train a machine learning model to achieve satisfactory performance on the whole dataset.

Conducting coreset discovery in a supervised fashion is fundamental, but unfortunately, it does not directly contribute to solving the problem of having to annotate all samples upfront Ju et al. (2022). Moreover, while the idea of supervised coreset discovery has received a lot of research attention for years (Olvera-López et al. 2010, Rolnick et al. 2017, Killamsetty et al. 2021, Barbiero et al. 2020), some studies recently are attempting to achieve similar results without the need of labels as a prerequisite, works like Ju et al. (2022) attempt to discover coresets in a fully unsupervised fashion.

While there are many different methods, one of particular interest in this work is the employment of evolutionary algorithms to approximate such coresets. This interest stems from the ability of these algorithms to allow for multi-objective optimisation, which, as noted by Barbiero et al. (2020), is of extreme importance since the approach at hand should balance the quantity as well as the quality of the chosen records.

To address this scenario, this thesis presents a novel evolutionary-based approach to coreset discovery by:

1. Adapting neuroevolutionary algorithm for neural architecture search to conduct large-scale image coreset discovery.
2. Proposing a new genome construction approach for evolutionary coreset discovery, which enhances the explainability of the results.
3. Presenting a novel unsupervised evolutionary-based approach for coreset discovery using convolutional autoencoders.

5.2.1 Coreset Discovery

There is a multitude of different methods for "dataset pruning" -as termed in Wang et al. (2018). In this work, dataset pruning is loosely used to describe the process of reducing the dataset in any way. These methods could be categorised into three main groups: 1. instance selection when part of the samples are selected and are deemed to be "valuable" (Olvera-López et al. 2010). 2. Active learning, when a portion of the data (usually the

part a particular model predicted with the lowest confidence (Killamsetty et al. 2021)) is used to estimate what new data needs to be labelled and used for training with a solid human-in-the-loop presence for progressively supplying new labels. 3. Dataset distillation, consisting of artificially generating samples that can trick a certain model into making sensible predictions on the normal data Wang et al. (2018).

Dataset distillation methods can compress a full dataset into a single instance per class. These methods have recently achieved impressive compression rates (Barbiero et al. 2019). However, producing these distillations requires significant time to train the model on the full dataset upfront (Wang et al. 2021). In addition, any changes to the model or additions to the data require recalibration of the distillations, which can be expensive and computationally intensive (Wang et al. 2018). Despite their effectiveness, the high cost of producing and maintaining dataset distillations may limit their practicality in certain contexts.

This study focuses on coreset discovery, a method of instance selection that aims to identify the most "valuable" or informative samples in a dataset Ju et al. (2022). Coreset discovery approaches can be grouped based on their feature extraction process and the use of labels Ju et al. (2022). This work does not consider methods that rely on manually specified features, such as those described in Campbell and Broderick (2018), Tsang et al. (2005), Tschitschek et al. (2014), Wei et al. (2015), as manual feature engineering is infeasible in the context of industrial (big data) applications for which MIRA-ME is intended.

As the coreset discovery field rapidly evolves, no standard established methods exist. Still, some of the most revolutionary and best-performing ones are presented here to the best of the author's knowledge. A popular approach a while ago was to use clustering methods such as k-means or similar techniques Har-Peled and Mazumdar (2004), Har-Peled and Kushal (2005). Later, the idea was refined by methods such as Frank-Wolfe Clarkson (2010), and GIGA Campbell and Broderick (2018) that are based on the comparison between approximated and full likelihoods.

However, findings in Toneva et al. (2018) pushed the field's boundaries even further by suggesting that coresets can be discovered using neural networks and examining closely catastrophic forgetting events. This "forgetting approach" quickly became popular and inspired multiple methods like Ju et al. (2022), Barbiero et al. (2019), Valvano et al. (2018) and the one presented in this chapter, which leverages the findings of Toneva et al. (2018).

Although many of the approaches seem to be reducing the datasets successfully, Barbiero et al. (2019) and Killamsetty et al. (2021) discuss the importance of phrasing the coreset discovery problem as a multi-objective optimisation, striving to minimise the used data points, while preserving performance as much as possible. An example of such bi-level optimisation techniques is Killamsetty et al. (2021) 's study - Glisten- which out-

performs the state-of-the-art by balancing model performance and coreset size through the use of Naive Bayes as well as nearest neighbour search in an iterative fashion.

As discussed in Chapter 2, evolutionary algorithms are powerful population-based stochastic optimisation techniques that are renowned for their ability to conduct multi-objective optimisation Deb et al. (2002a), Zhang and Li (2007), Deb (2014).

At the early stages of the "instance selection" field, works tried to utilise these powerful tools Derrac et al. (2012). However, to the best of the author's knowledge, the only work that has successfully utilised evolutionary algorithms to address coreset discovery in the context of a somewhat large-scale dataset is EvoCore Barbiero et al. (2019). They have demonstrated that evolutionary algorithms (NSGA-II Deb et al. (2002a) in particular) can be used to extract coreset from many problems, including Mnist, a low-dimensional computer vision dataset.

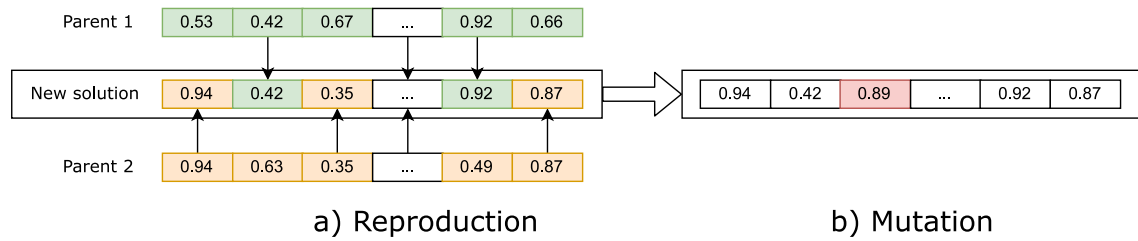
Approaches discussed so far fall in the family of supervised coreset selection. However, new ways to conduct coreset selection have been recently proposed (Valvano et al. 2018, Ju et al. 2022). This new niche uses *contrastive* learning and the ability of autoencoders to compress and reconstruct images out of these compressions. Specifically, the idea behind (Valvano et al. 2018) is that an objective function can be defined such that it takes into consideration how well a variational autoencoder can reconstruct images and how spread the representations of the data points are in the compressed space generated by the bottleneck layer. Along these lines, Valvano et al. (2018) argues that samples that can be generated through simple augmentation should not be part of the generated coreset and that they are "not valuable". To avoid picking such samples, they evaluate local manifold Euclidean distances (Valvano et al. 2018).

By contrast, the work of (Ju et al. 2022) focuses on the cosine similarity of samples to discriminate representations of data for selecting coreset candidates. Their framework utilises SimCLR Chen et al. (2020), MoCo (He et al. 2020) and others to showcase the potential of using contrastive learning for coreset selection.

Unsupervised coreset selection studies consistently argue that the key limitation of supervised approaches is the requirement for labelled data (Valvano et al. 2018). In contrast, unsupervised approaches do not rely on labelled data and can, therefore, be more widely applied (Ju et al. 2022). Thus, unsupervised coreset discovery has become a promising area of research, with the potential to enable more efficient and effective instance selection in a variety of contexts (Valvano et al. 2018, Ju et al. 2022, Barbiero et al. 2019).

The authors of Barbiero et al. (2019) employ a simple ridge classifier instead of a computer vision non-linear model such as a convolutional neural network. Their work is leveraged as a stepping stone to the initial design of the MIRA-ME backbone. Specifically, in their study, authors phrase the evolutionary problem as a discrete search of image indices, despite evolutionary algorithms' tendency to operate better with continuous

Figure 5.7: MIRA-ME variation process. Notice that during the variation, the selected coreset encodings reproduce, and then variation is applied for the effective exploration and exploitation of the search space.



variables (Rothlauf 2006, Larranaga et al. 2013).

5.2.2 Methodology

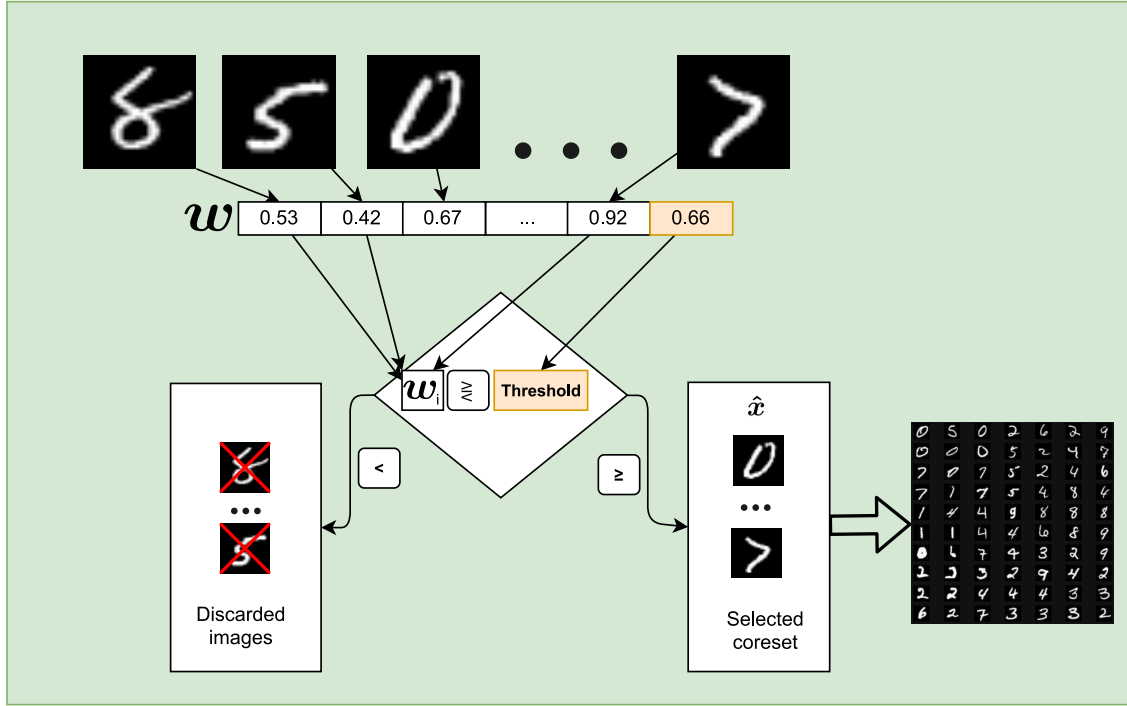
Here, each sample (an image denoted by the tensor x_i) is assigned a continuous weight while letting the algorithm control the threshold above which indices can be selected. This filtering contrasts with previous evolutionary strategies, which instead phrase the task such that the algorithm samples indices in a discrete fashion Barbiero et al. (2019).

The advantage of MIRA-ME is that it conserves information about the importance of a certain sample within the population and propagates the joint progress for all solutions through the evolutionary processes (genome selection and variation). Even further, the variation is performed in a more intuitive way with a continuous search space instead of a discrete one like previously Barbiero et al. (2019).

A genome w (out the population \mathbb{W} of possible genomes) is simply an $n+1$ -dimensional vector of continuous values $w_i \in (0, 1]$, $i = 1, \dots, n+1$, where n is the number of images (samples). Each vector entry $w_i, i \leq n$, represents a weight assigned to the corresponding image (initially random), which encodes the importance of the image for the classification task and hence its susceptibility to being part of the coreset. The last entry, w_{n+1} , encodes a threshold value.

To discover critical datasets, MIRA-ME relies on two different methods- the "naive" pipeline and the "unsupervised" pipeline. Both pipelines rely on already established evolutionary instance selection methodology (Derrac et al. 2012) and an encoding (w) discussed above, as well as variation based on Figure 5.7. Adding the threshold to be part of the genome leaves both the sample selection and determining the threshold to the algorithm, which lets the algorithm fully control the instance selection process in difference to previous works (e.g., (Barbiero et al. 2019)). This unique property is valuable since the size of the coreset is not known in advance and methods like Glister (Killamsetty et al. 2021) or SimCLR (Chen et al. 2020, Ju et al. 2022) share the limitation that all of them require it as a hyperparameter.

Figure 5.8: MIRA-ME images encoding process. Notice that each image x_i has an associated weight in the genome w and the threshold (w_{n+1}) used to determine if an image is "important enough" to be in the coreset is part of the genome and is thus subject to the discovery process of the evolutionary algorithm.



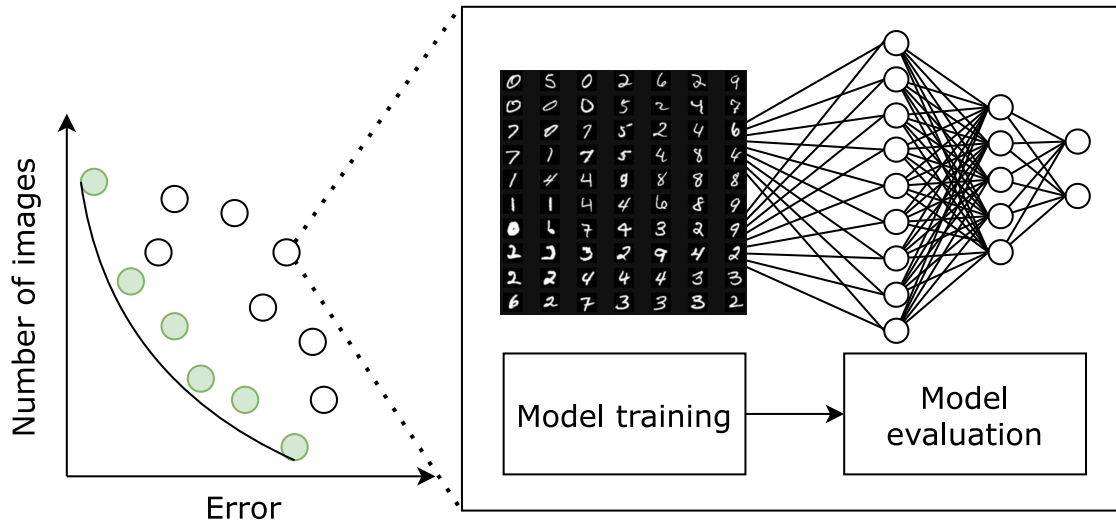
The way the image candidates are chosen to belong to the coreset during each iteration in both the naive and the unsupervised pipelines is by simply selecting each i^{th} image x_i whose weight w_i exceeds the threshold w_{n+1} (illustrated in Figure 5.8). The naive pipeline then uses the coreset as a classification task in a multi-objective setting, providing the output y containing the values of the multiple objectives (Figure 5.9). The first objective y_1 (the first entry of y) is just the current number of images in the coreset,

$$\begin{aligned} \underset{w \in \mathbb{W}}{\text{minimise}} \quad y_1(w) &= \sum_i^n \hat{w}_i, \text{ where } \hat{w}_i := \begin{cases} 1 & \text{if } w_i > w_{n+1} \\ 0 & \text{otherwise,} \end{cases} & (5.6) \\ \text{subject to } w_i &\in (0, 1] \forall i \in \{1, \dots, n+1\} \subset \mathbb{N} \end{aligned}$$

The second objective (y_2) in the naive pipeline is the validation performance obtained after training for 5 epochs (unless specified otherwise) using the provided model and the selected coreset of images \hat{x} , which represents an $\hat{n} \times 1$ set of coreset images x_i . Interestingly, every "weight" w_i in the genome can be considered as a proxy of the importance of its corresponding image (x_i), and hence its proneness to belong to the coreset. This property fosters explainability of the produced coreset \hat{x} .

Even though the naive approach seems to be working well in literature (Chen et al. 2020, Ju et al. 2022, Barbiero et al. 2019), MIRA-ME goes one step further by compress-

Figure 5.9: "Naive" MIRA-ME evaluation process. The two objectives considered here are the number of images and the task error rate. Notice that the green dots (representing non-dominated solutions) are the solutions that are going to be selected during the selection process based on their contributing hypervolume. As part of the evaluation, the network is trained with the selected coreset \hat{x} for each solution in the population (dot in the scatter plot on the left).



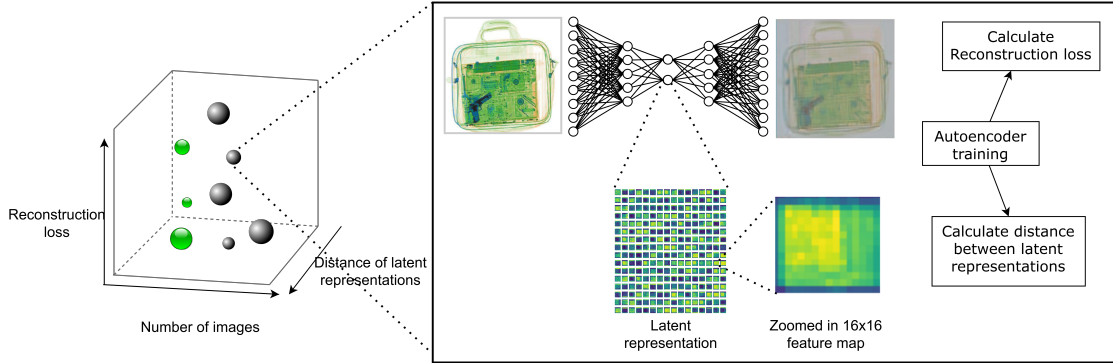
ing the dataset in an unsupervised fashion. This is of critical importance in scenarios where labelling data or data collection is expensive. Using this unsupervised pipeline, it is possible to filter out highly representative data and then focus on it or extract valuable insights about the dataset at the early stages of data collection before labels are provided. The beauty of not needing labels is that a task may not be defined yet or it may be changed down the line, and it is not a prerequisite for applying the unsupervised pipeline (Valvano et al. 2018).

The unsupervised pipeline (the evaluation process of which is presented in Figure 5.10) operates in the following way: First, an autoencoder (parametrised by θ) is constructed ($\hat{z} = f(\hat{x}; \theta)$), which is composed of two parts. The encoder part, $\mathbf{a} = g(\hat{x}; \psi)$ with parameters ψ is such that each row coreset \hat{x} is transformed and compressed in a smaller set of activations \mathbf{a} in the bottleneck layer, where \mathbf{a} represents a $\hat{n} \times 1$ set containing the compressed representations a_i of each coreset image x_i (for simplicity in the description, this summarised representation was used for all images).

Then, compressed representations \mathbf{a} are used to reconstruct an approximation of the raw images $\hat{z} = h(\mathbf{a}; \phi)$ where h (parameterised by ϕ) is the decoder component of the autoencoder f .

The autoencoder is trained for 10 epochs, and then activations \mathbf{a} from the bottleneck layer are extracted to compute their $\hat{n} \times \hat{n}$ covariance matrix C (where \hat{n} is the number of images in the coreset \hat{x}).

Figure 5.10: Evaluation in the "unsupervised" pipeline. Here there are 3 objectives: number of images, reconstruction loss and distance of latent representations. The latent representations are captures from the trained convolutional autoencoders used for the particular dataset. The autoencoders used here are discovered using MONCAE. Notice that in the sample provided, the threat is clearly visible in the reconstructed image.



Then, the Mahalanobis distance of each bottleneck layer activation a_i corresponding to the image \hat{x}_i (McLachlan 1999) is calculated, which has proven to be effective with multi-dimensional data problems (Gallego et al. 2013):

$$d_i = \sqrt{(a_i - \mu \cdot \mathbf{I})^T \cdot C^{-1} \cdot (a_i - \mu \cdot \mathbf{I})} \quad (5.7)$$

where μ is the overall mean of the covariance matrix and \mathbf{I} is a column vector of ones of the same dimensionality as the compressed representation of the i^{th} image, a_i (note that $p \times p$ bottleneck layer activations to was transformed to $2p \times 1$ column vector for performing this operation). Thus, d_i act as a proxy (under normality assumptions) of the likelihood of the compressed representation of the image \hat{x}_i .

The final step is to calculate the average \bar{d} of all distances, defining the first objective in the evolutionary algorithm as $d_i - \bar{d}$, (informally, the "spread" of the average Mahalanobis distance for the i^{th} image).

The second objective is how well the autoencoder performed on a held-out validation set by minimising the mean squared error between the samples from the validation set and their \hat{z} counterparts. The third objective is the number of samples (like in the naive approach). In short, a major advantage of both MIRA-ME pipelines over other state-of-the-art (e.g., (Killamsetty et al. 2021)) is that the coreset size is dynamically discovered by the algorithm itself (there is no need to specify it upfront). Thus, no prior knowledge of the data itself is required and hence the process can be fully automated. In addition, instead of running multiple times the algorithms in an effort to understand performance with different compression rates (e.g., Killamsetty et al. (2021), Ju et al. (2022)), MIRA-ME generates a pool of solutions (approximation set) that to select from, and hence no manual tuning is required.

5.2.3 Experimental setup

To evaluate how effective both the "naive" and the "unsupervised" pipelines are, their performance is compared to random sampling as well as to state-of-the-art methods for coreset discovery. Towards this goal, unless otherwise specified, the average results of five independent runs with the selected method/coreset size are presented. Interestingly, even with the uniform random selection for both MNIST and CIFAR-10 no high standard deviation is observed and the results of the separate runs have minimal standard deviation (less than 2%), hence they are considered highly representative of the problems.

The experiments are conducted using three separate datasets that were already used as part of the previous chapters. Moreover, some of the few computer vision coreset approaches in literature use these datasets, allowing MIRA-ME results to be comparable to the state-of-the-art. Below, some important properties in the context of this particular work of the already discussed datasets are established:

1. MNIST - a dataset containing 70000 visual 28x28 grayscale images of all 10 digits. 10000 of them are used as a held-out test set.
2. CIFAR10 - a dataset containing 60000 visual 32x32 coloured images of 10 different classes mainly composed of animals and vehicles. 10000 of them are used as a held-out test set.
3. SIXray10 - a dataset containing 74960 x-ray high-resolution images. This dataset is multi-label, meaning that even though there are five classes representing different types of weapons, each image can belong to a one, many or none of the classes. This multi-label problem makes 32 possible unique label combinations. There is also a huge imbalance present in the different classes, which is why accuracy is not a good measure for this dataset (Miao et al. 2019).

To keep the results as comparable as possible LeNet (LeCun et al. 1998) is used for both MNIST and CIFAR-10 ¹. For SIXray, LeNet is unable to capture the dataset well, making deeper ResNet-50 a viable option (He et al. 2015). The models are trained for the same number of epochs using SGD optimiser and the same schedule for the learning rate. The models are trained with no augmentations to ensure the results are as controlled as possible and to enhance the reproducibility of the results.

With CIFAR-10, the same methodology is followed, however to ensure all models are trained in the same way for, a specific training cycle is used ² where instead of using the full CIFAR-10 dataset only the samples discovered by the evaluated method are used. For the benchmarks, 10% and 20% of the dataset are used compared to 0.1%, 1% and

¹With the exception of EvoCore and SimCLR for CIFAR-10

²defined by the following repository <https://github.com/exelban/tensorflow-cifar-10>

10% for MNIST. To implement Glister, SIMCLR and Random sampling, the DeepCore (Guo et al. 2022) is used in line with the aim of keeping as many control variables as possible.

One of the aims of MIRA-ME is to investigate the performance of coresets methods on large-scale data. While most of the literature focuses on small, low-dimensional datasets (such as MNIST, CIFAR-10, etc.) for proof of concept (Valvano et al. 2018, Barbiero et al. 2019, Killamsetty et al. 2021, Ju et al. 2022), it is essential to establish benchmarks and evaluate the feasibility of these methods. However, some approaches are designed in a way that makes them incompatible with real-world problems (Ju et al. 2022). The aim is to demonstrate the utility of MIRA-ME in large-scale problems. To that end, SIXray is used as the dataset for the final set of experiments. Unfortunately, as SIXray is a multi-label problem, it is incompatible with any of the unsupervised approaches tried. Therefore MIRA-ME is compared to random selection and Glister. As this dataset has a high-class imbalance, accuracy is not the best metric to assess performance, so average precision is used for each class and the mean over all classes. The same training cycle for all methods is used, training for 50 epochs using the SGD optimiser with a learning rate of 0.001 and a learning rate scheduler available in the MIRA-ME repository. ResNet50 is used as the architecture. Here, for random and Glister use 3 budgets are used - 10%, 15% and 20%.

To discover a suitable autoencoder for the unsupervised pipeline, MONCAE (presented in Chapter 3) is employed. The default hyperparameters are used for the discovery of the autoencoder, and the best-performing autoencoder (in terms of final cHV) is then used.

5.2.4 Results and Discussion

Turning now to the results, Figure 5.11 shows the performance of various methods on the MNIST dataset, which is the lowest-dimensional dataset used in the selection. Both random sampling and Glister are used with 10% (6000 samples) and 1% (600 samples) of the dataset. While most studies focus on budgets of 10% to 70% of the data, the focus here is on the low-data regime and how the algorithms can compress the datasets the most while retaining the original performance as much as possible. For the MIRA-ME experiments, the ccoreset with the highest assigned contributing hypervolume indicator of the produced set of solutions is used. In the MNIST scenario, the ccoreset with maximum achieved compression is denoted by "mc" in the results table.

As expected in the 10% budget, Glister(Killamsetty et al. 2021) and SimCLR(Ju et al. 2022) outperform random sampling, with SimCLR achieving the best MNIST result in the whole selection. However, in the low-data regime (1% budget), it appears that random sampling performs better at finding representative samples, as indicated by the results in the table. This can be attributed to the fact that the distribution of classes in MNIST is

Figure 5.11: MNIST MIRA-ME results. Random, Glister and SimCLR are run with 10% and 1% of the dataset, and Random is run with 0.1%. Notice that in the ultra-low data regime (below 1%), Random is clearly outperformed by genetic algorithms such as EvoCore and MIRA-ME with maximum compression (MIRA-ME (naive) mc). MIRA-ME (naive) mc stands for the coreset of the final population that has achieved the maximum compression. Also, notice that both the naive and unsupervised versions of MIRA-ME outperform Random with significantly fewer samples. While SimCLR and Glister beat them, they use over 1000 more samples (and over 4000 in the case of the unsupervised pipeline) than MIRA-ME.

Coreset Method	Number of samples	Test Accuracy
Random	6000	95.68%
Random	600	91.90%
Random	60	66.81%
Glister	6000	98.53%
Glister	600	64.50%
SimCLR	6000	98.13%
SimCLR	600	84.44%
EvoCore* ³	82	77.2%
MIRA-ME (naive)	4918	97.67%
MIRA-ME (naive) mc	70	79.58%
MIRA-ME (unsupervised)	1859	95.70%

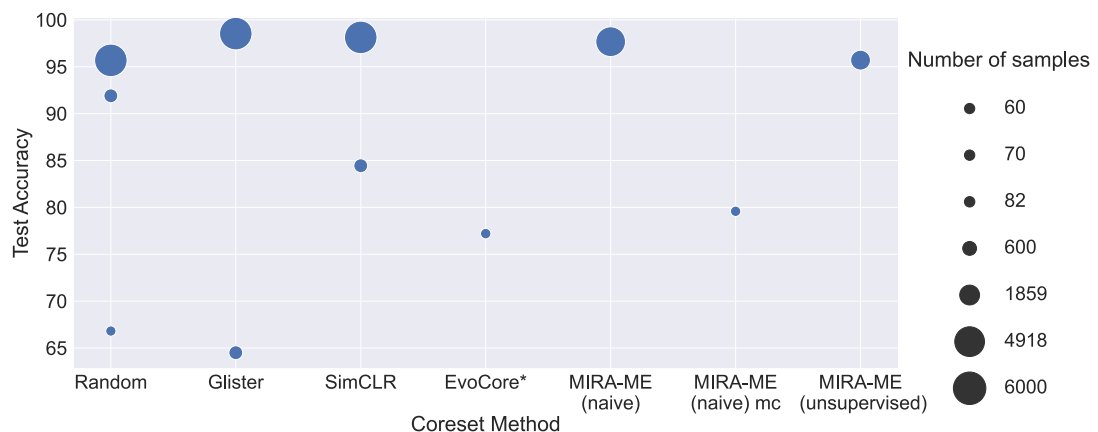


Figure 5.12: MNIST pixplots

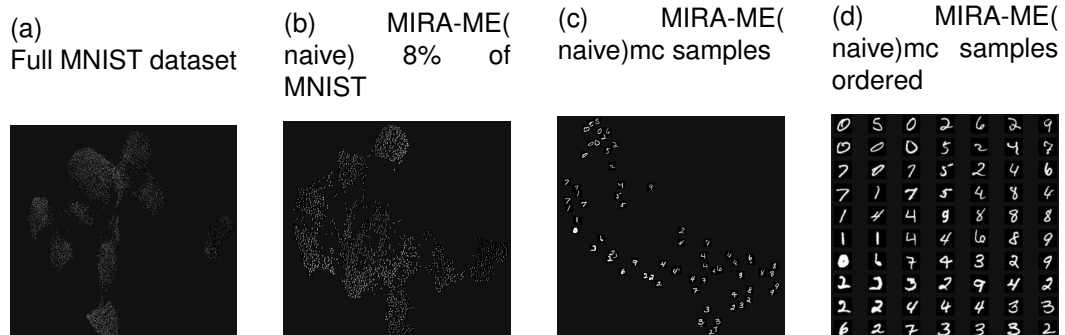
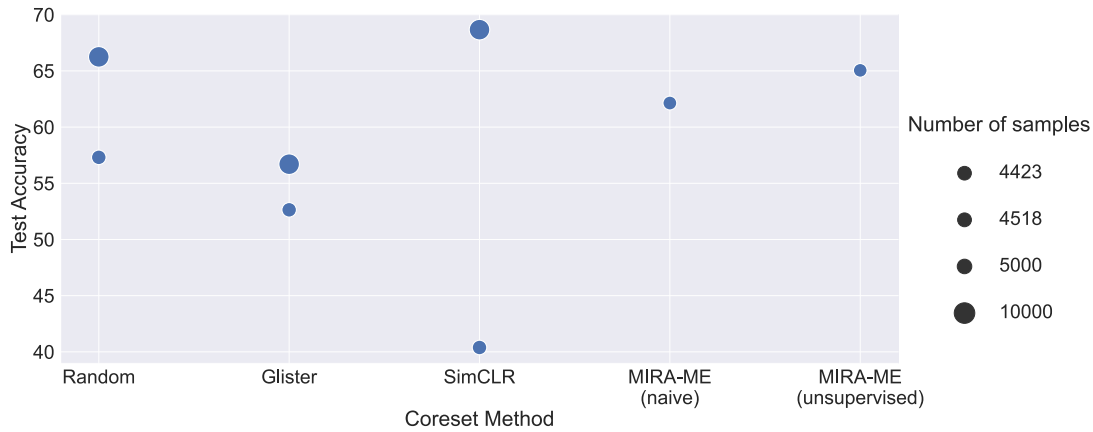


Figure 5.13: CIFAR-10 MIRA-ME results. Notice that the advantages of MIRA-ME here are even more prominent. While both pipelines are slightly outperformed by the other approaches when they use more than 2x the data, both MIRA-ME pipelines (using less than 8% of the data) outperform all approaches when they use 10% of the data. Interestingly, the unsupervised pipeline here is again noticeably stronger than the naive pipeline and uses even fewer samples.

Coreset Method	Number of samples	Test Accuracy
Random	5000	57.31 %
Random	10000	66.25%
Glister	5000	52.64%
Glister	10000	56.70%
SimCLR	5000	40.39%
SimCLR	10000	68.67%
MIRA-ME (naive)	4518	62.14%
MIRA-ME(unsupervised)	4423	65.05%

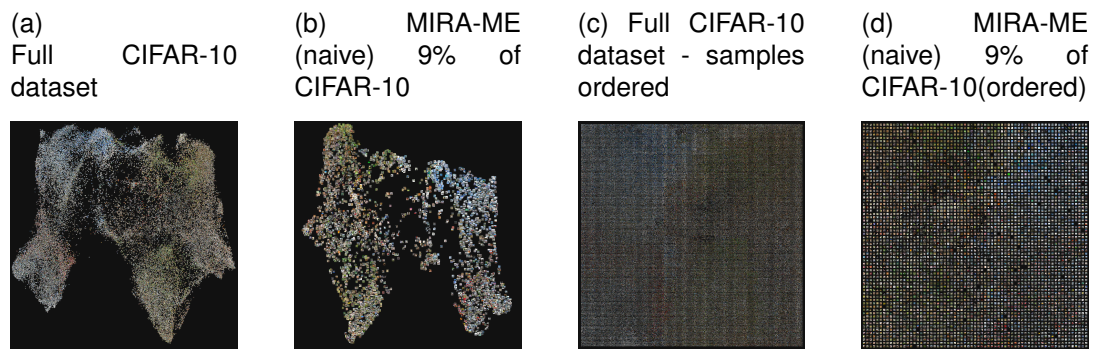


uniform; thus, random does have a high implicit chance of picking good samples, whereas Glister’s way of picking “hard samples” can lead to getting too many of a certain class and too little of another at this high compression rate. This is in line with the experiments of the authors when they compare Glister to Random with and without class imbalance (Killamsetty et al. 2021). SimCLR results in the low data regime are consistent with the ones of Glister, and the fact that the method only has one objective (cossim pair distance in this case) might hinder its usefulness for such uniformly distributed datasets.

Nevertheless, MIRA-ME manages to beat the 10% Random while using a little over 8% of the training data and with the “unsupervised” pipeline, it is on par with it while using only 3% of the data. This performance can be credited to the fact that both MIRA-ME approaches have multiple objectives instead of just one. EvoCore³’s results (Barbiero et al. 2019) are also included as their work was a major motivation to attempt to use evolutionary algorithms to start with. Even though EvoCore uses the Ridge classifier instead of LeNet, the results are a testament to the potential of utilising evolutionary

³EvoCore uses a Ridge classifier and is not designed to work with deep learning models. Thus, it is not directly comparable to the other results generated with the same methodology.

Figure 5.14: CIFAR-10 pixplots



algorithms in these scenarios.

While Glister and SimCLR achieve better accuracy at 10%, MIRA-ME uses less data and also provides a set of solutions. Moreover, in the low data regime, MIRA-ME outperforms GLISTER significantly by beating the 1% accuracy of Glister with only 70 samples (0.1%). Even though MIRA-ME achieves better accuracy with fewer samples than EvoCore, the comparison between the two is not entirely fair since EvoCore uses a linear classifier instead of a CNN. The best part of the MIRA-ME results is that the assigned scores for every single image provide an extra layer of explainability. While the discovered threshold is used to find the actual subset, these scores can be used separately. Low as well as high-scoring samples can be inspected for potential noise or other insights.

Interestingly, the "unsupervised" pipeline achieves almost the same accuracy as the "naive" one while using even fewer samples. In general, a trend begins to emerge that the unsupervised approaches are generally getting better scores than their supervised counterparts. This finding may feel counterintuitive at first as the provided labels in the supervised experiments should provide valuable information that the "unsupervised" pipeline doesn't have access to. On the other hand, reconstructing the image might inherently improve the representations learnt by the network and foster the idea of "concept" representations more closely.

To put this into perspective, in Figures 5.12 and 5.14 (with more detailed results view in Figure 5.13) and 5.15, a library called PixPlot⁴ is used to visualise the different sets. Looking at Figure 5.12a and 5.12b, the compression becomes apparent because, on the former, no digits are visible, where on the latter then start to become recognisable. In Figure 5.12c and 5.12d, the difference with the full dataset is even more obvious and the whole dataset can be easily displayed even with the constrained space in this A4 sheet of paper.

What stands out in Figure 5.13 is the strong random baseline, which manages to outperform Glister both at the 10% and 20% marks. This discrepancy with the results

⁴<https://github.com/YaleDHlab/pix-plot>

Figure 5.15: SIXray10 pixplots

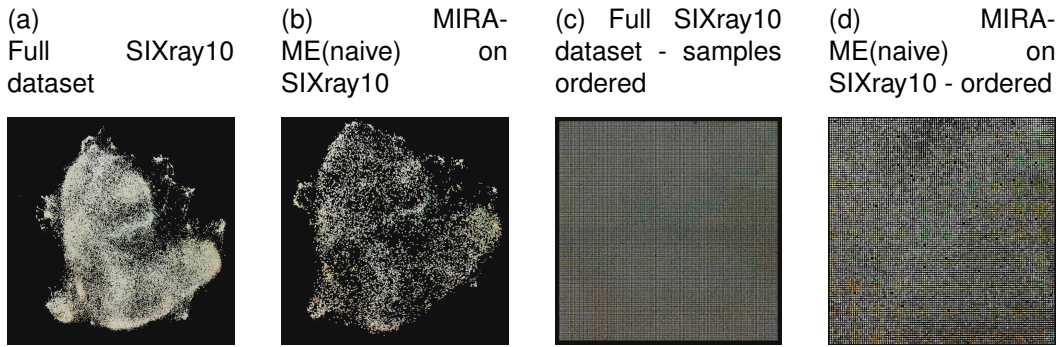


Table 5.5: SIXRAY 10 MIRA-ME results. Notice that the MIRA-ME approaches are better than both Glister and Random, even when these approaches use up to 3x more data. What is more, the 62% achieved by the unsupervised pipeline (using less than 5% of the data) is close to the model’s performance with the full dataset of around 77% presented at the beginning of the chapter.

Coreset Method	# of samples	Gun	Knife	Wrench	Pliers	Scissors	Average
Random	7496	71.41%	46.75%	15.68%	28.95%	0.08%	32.57%
Random	11,244	71.26%	36.72%	18.75%	42.92%	8.00%	35.53%
Random	14,992	69.15%	50.02%	33.73%	31.08%	0%	36.80%
Glister	7496	77.72%	34.46%	24.35%	50.74%	5.71%	38.60%
Glister	11,244	82.07%	60.34%	24.03%	49.90%	11.11%	45.49%
Glister	14,992	84.13%	53.70%	23.08%	54.06%	18.45%	46.68%
MIRA-ME (naive)	4918	85.00%	62.16%	29.96%	61.71%	36.36%	55.01%
MIRA-ME(unsupervised)	3536	70.64%	68.52%	28.50%	49.15%	93.33%	62.03%

presented in Killamsetty et al. (2021) can be attributed to several factors. Firstly, LeNet is used instead of ResNet (used in Killamsetty et al. (2021)). Even though MNIST and CIFAR-10 are used, results consistently show the substantially lower overall capacity of LeNet might fail to capture the underlying distribution, which can result in poorer manifold representations.

Consequently, the used manifold representations might affect the ranking of Glister and ultimately cause overall poorer performance, which highlights a potential limitation of Glister. Secondly, CIFAR-10 is uniformly distributed in terms of classes. The smaller portion of the dataset used compared to the original study might further reveal a caveat of Glister in low data regime. Another possible explanation is that no data augmentation or complex pre-processing are used during training which differs from the experimental setup in Killamsetty et al. (2021) for the purpose of keeping as many control variables as possible. Possibly, the use of augmentations helps Glister build robust representations and then pick from them more successfully.

SimCLR appears to perform poorly in the low data regime (10% budget) but shows significant improvement when the budget is increased to 20% and achieves the highest test accuracy among all methods. Notably, both the "naive" and "unsupervised" versions

of MIRA-ME outperform all methods in the 10% budget, with the "unsupervised" version demonstrating a clear advantage. It not only uses the smallest number of samples but also significantly outperforms the benchmark methods and competes with their 20% budget scores. In contrast to MNIST, where the contributing hypervolume for the "unsupervised" version favoured size compression, the number of samples chosen by the "unsupervised" version in this scenario is similar to the number chosen by the "naive" version.

Compared to Glister, MIRA-ME("unsupervised") appears to be more effective at selecting critical samples in the low data regime, likely due to its use of an additional objective that allows it to utilise not only the contrast between different samples⁵, but also the ability of the autoencoder to reconstruct images. In Figure 5.14, the original full dataset is visualised alongside MIRA-ME's more than 90% compressed version. The visualisation reveals that the general trends of the features in the dataset are preserved and that samples with distinctive features are more likely to be selected than those that are closer to the dataset's mean and standard deviation values in terms of pixel values.

Turning now to the real-world large-large applications, Table 5.5 displays the experimental data on SIXray. As seen from the table, MIRA-ME's best candidates feature sets that are close to the 5% mark while achieving unprecedented performance and score close to some methods with the full dataset presented in (Miao et al. 2019) and in Section 5.1. Strikingly, MIRA-ME's unsupervised pipeline once again has the lowest number of samples (even lower than the one in the CIFAR-10 experiments), yet it manages to outperform significantly all of the other approaches with an average precision of 62%, which is almost double the one of random with 20% of the data and comfortably ahead of Glister with substantially less used data also.

The "naive" approach is lagging behind with an overall of 55%, but it can be observed that the unsupervised approach has really focussed on the most underrepresented class (the Scissors), and it has achieved better test precision than even specifically tailored approaches working with the full dataset (Miao et al. 2019). Here, the benefits of Glister over the random selection at each budget are also observed, as well as how random fails at picking up enough scissors for the algorithm to learn what they are as a whole.

Surprisingly, the unsupervised pipeline turns out to be the better one of the two MIRA-ME approaches, but it has its own drawbacks, which should be explored in future work. For example, the prerequisite for a good autoencoder may be considered a severe bottleneck, and while it can be trained in an unsupervised fashion, it does increase the overall training time of the approach, which may be important in certain use cases. These processes, when combined with approaches like MONCAE, have demonstrated promise to automate streamlining optimisation of both model and data fully.

⁵(measured using the distance between autoencoder bottleneck representations as described in Section 5.2.2, rather than cosine similarity distance used by SimCLR)

5.2.5 Conclusion and Future work

As the problem of finding coresets has become increasingly popular (Har-Peled and Mazumdar 2004, Dubey et al. 2018, Barbiero et al. 2020), in this work, a novel evolutionary-based approach is presented. It is tasked and manages to achieve a state-of-the-art balance between the compression rate and the performance of the models on multiple datasets. Both a supervised ("naive") as well as an unsupervised variations of the approach are showcased. In difference to the state-of-the-art discussed in Sections 5.2.1 and 5.2.4, both variations automatically determine the size of the coreset. Also, the user is presented with the option to choose from a variety of different solutions and extract further insights for the data, such as the individual score per image.

Firstly, the "naive" approach works in a supervised fashion to discover the samples that can supply enough information to train a model to perform well on the defined task, while the "unsupervised" version features a fully unsupervised approach that only requires data and a few hyperparameters such as number of generations and population size.

In the experiments, the two approaches achieve the highest performance compared to the alternatives while keeping the coresets the smallest. Surprisingly, the fully unsupervised approach does beat the supervised one, which is contradictory to the author's initial intuition.

In addition, this is the first work, to the best of the authors' knowledge, to attempt to use coreset methods on large-scale datasets such as SIXray. What is more, no previous studies, to the best of the authors' knowledge, explore the use of compatible coreset methods for multi-label problems. Problems such as SIXray is where the benefits of MIRA-ME seem to shine the most.

A limitation of this work is the lack of extensive ablation studies that determine the effect of different hyperparameters of the evolutionary algorithm (such as the population size, number of generations and others), which should be explored by future work together with the causality of how well the convolutional autoencoder performs before the selection in the "unsupervised" pipeline.

Future work should also explore the use of all mentioned approaches in other large-scale datasets. Even though MIRA-ME has attempted to go into low data regime modes with the sub 10% coresets, it is evident that there is significant room for improvement for going below 1% and even less. What is more, no techniques that employ augmentations of the discovered coresets were explored, which might be crucial for further reduction.

Chapter 6

Concluding Remarks and Future Work

This thesis provides a stepping stone towards more widespread use of Automated Machine Learning (AutoML) in research and industry. Specifically, the focus is on fostering the efficiency of AutoML approaches by significantly reducing their computational cost while preserving or improving their effectiveness (as discussed in Chapter 1). Methods are designed with a holistic perspective in mind, that is, for addressing fundamental computer vision problems. Thus, they are evaluated in various fundamental datasets and compared with state-of-the-art approaches. After this, the thesis concentrates on adapting such techniques to novel applications for threat identification.

6.1 Thesis summary and main contributions

The first objective of the thesis **Objective 1**¹ is achieved in Chapter 2 with a comprehensive review of the current state-of-the-art in AutoML for computer vision, highlighting both the key strengths and limitations of these approaches. Emphasis is placed on discussing the various methods used in concealed threat identification and the general working principles of computer vision models. This analysis aims to identify future research and development areas in AutoML for computer vision. Previous AutoML algorithms proved to be effective in discovering models that outperform those designed manually by humans (e.g., (Real et al. 2017, Qin and Wang 2019)). However, it becomes evident that new, significantly more efficient systems are needed to progress the AutoML and Neural Architecture Search (NAS) fields. In particular, the use of the current state-of-the-art in industrial applications is still unfeasible mainly due to their tremendous requirement for computational resources even for small-scale toy classification datasets (e.g., Stanley

¹Identify key strengths and limitations of the state-of-the-art in AutoML for computer vision, with an emphasis on neuroevolution and threat detection.

and Miikkulainen (2002b), He et al. (2021)).

Next, Chapter 3 proposes an efficient AutoML approach for the discovery of convolutional autoencoder architectures (MONCAE), capable of simultaneously optimising image reconstruction, dimensionality reduction and model complexity in line with **Objective 2**². Some of the key findings of this chapter are:

1. **The feasibility study(Section 3.1):** A feasibility study has shown that the use of convolutional autoencoders can improve the quality of extracted features through the incorporation of an additional self-supervised step in the training cycle of CNNs. This step utilises convolutional autoencoders and transfer learning to enhance final task performance. While the gain of 2% may not appear significant at first glance, statistical analysis has determined that it is, in fact, statistically significant. The results suggest that using convolutional autoencoders as a supplement to the conventional training of CNNs may be a viable approach to improving feature extraction and task performance. Also, the results demonstrated that convolutional autoencoders could be used prior to receiving labels, which in some industrial settings can take a long time (Goodfellow et al. 2016b).
2. **The novel neuroevolution algorithm:** A novel neuroevolution algorithm that automatically discovers convolutional autoencoders for arbitrarily supplied dataset while performing multi-objective optimisation (MONCAE) achieved over $10\times$ compression while outperforming state-of-the-art (including Charte et al. (2020)). With 97%, close to 88% and 62% classification accuracies on MNIST, Fashion-MNIST and CIFAR-10, respectively, MONCAE's autoencoders outperform other methods as evident from Table 3.3, Table 3.4 and Table 3.5.
3. **The novel metric:** A novel metric and objective for successful convolutional autoencoders, named "level of compression", was introduced as part of Section 3.2, measures how the scale of the bottleneck layer compares to the input dimensions. The score was inspired by the use of network parameters as a proxy for the complexity of the model in other works such as Hinton et al. (2015) and Charte et al. (2020).
4. **The results:**MONCAE achieves promising results on small-scale datasets but also presents the potential of using neural architecture search and neuroevolution within a limited time (for instance, 93 GPU minutes compared to 1440 for MNIST in Charte et al. (2020)), by optimally balancing image reconstruction, dimensionality reduction and model complexity, w.r.t. the closest previous attempts (e.g. 97.3% accuracy with a LOC of 1.8 compared to 96.3% accuracy with a LOC of 2.2 for (Charte

²Design an efficient AutoML approach for computer vision, capable of simultaneously optimising multiple objectives (image reconstruction, dimensionality reduction, model complexity) to discover convolutional autoencoder architectures.

et al. 2020)). The comparison with the state-of-the-art in MONCAE was challenging since it signifies the first attempt to date (to the best of the authors' knowledge) to construct convolutional autoencoders automatically.

Next, to address **Objective 3**³ (and partially **Objective 5**⁵), Chapter 4 builds upon the promising results of MONCAE and introduces a novel Resource-Aware Multi-Objective Semantic Segmentation neuroevolution approach (RAMOSS). Some key takeaways from RAMOSS are:

1. **The encoding:** To address some of the key limitations of MONCAE, RAMOSS sets out to present a more optimal encoding of convolutional neural networks is constructed that is designed to work with segmentation models and optimised to scale to arbitrary data. The search space is also exponentially increased by designing an encoding capable of representing any connection between layers. The new encoding allows for each layer to be connected to each other layer through the use of a directed acyclic graph constructed by encoding a row of a binary upper triangular matrix as part of the encoding of each layer as explained in Section 4.1. The encoding is compatible with a plethora of different evolutionary algorithms to act as controllers of the process, and the framework is open-sourced with the invitation to researchers and practitioners to make use of RAMOSS with their data, experiment with hyperparameters and improve the provided stepping stone towards the transition to fully automated machine learning.
2. **The Progressive Stratified Split (PSS):** A new strategy is introduced as a novel way to speed up AutoML in semantic segmentation as part of Chapter 4. The PSS works by progressively iterating over stratified sub-samples of segmentation or multi-label data. The importance of PSS stems from its unique property to conserve architecture performance ranking, which makes the acceleration of AutoML possible (refer to Section 4.1.3).
3. **The results:** RAMOSS discovers architectures that are slightly worse than the top state-of-the-art in terms of validation performance (1-2% worse on CIFAR-10 and 3-5% worse on Cityscapes), but it uses 20-40 times fewer parameters and is one of the few AutoML approaches (Zoph and Le 2016, Real et al. 2017, Lu et al. 2019). Compared to state-of-the-art architectures of similar size (Howard et al. 2019, Shaw et al. 2019), RAMOSS outperforms them (with up to 4% on Cityscapes) while still having fewer parameters. RAMOSS also discovers its architecture in a record 0.3-0.4 GPU days, which, compared to the state-of-the-art, is a 30x-1000x+ improvement (Chen et al. 2017, Liu et al. 2019a). RAMOSS achieves the best trade-off,

³Develop a novel multi-objective optimisation AutoML strategy of discovering flexibly connected convolutional networks for semantic segmentation which can dynamically adjust to computational requirements.

which is established by using the cHV over the generated results, as evidenced by the tables in Section 4.3.

Overall, Chapter 4 confirms the conjecture from Chapter 3 that multi-objective neuroevolution can aid the discovery of optimal architectures for large-scale problems tremendously. Moreover, the training can be accelerated through heuristic methods such that the architectures can be utilised not only for toy problems, as in previous works (e.g. Real et al. (2017)), but also for real-world ones.

With the following chapter - Chapter 5 the thesis explores **Objective 4**⁴ and **Objective 5**⁵. In this study, the utility of RAMOSS and MONCAE, two approaches presented in previous chapters, is evaluated in the context of industrial applications. The results support the hypothesis that these approaches are not only effective at discovering optimal architectures(with MEOw), as they were designed to do but also demonstrate potential in the discovery of data space and coresets in particular(with MIRA-ME). In summary, the core findings and contributions from this chapter are:

1. **MEOw**: An improved version of RAMOSS is presented that uses state-of-the-art proxy scores to estimate the performance of an architecture without the need for training. The approach is named MEOw, although in Chapter 5 it is referred to as RAMOSS interchangeably since the MEOw represents the application of RAMOSS to the problem of threat identification and the modifications to achieve the acceleration, while important, exploit most of the concepts defined in RAMOSS. In this chapter, a way to utilise the benefits of choosing a population-based method is also presented. An ensemble method draws on a larger portion of the produced approximation set of solutions from the evolutionary algorithm, and exploiting their unique representational capabilities, the method beats current state-of-the-art methods for SIXray and demonstrates the value in producing multiple different solutions for a particular problem rather than relying on a single one. With MEOw, it is established that AutoML computer vision algorithms(such as RAMOSS and MONCAE) can be aided by proxy performance estimation scores that can accelerate the algorithms to run within a budget of 1 GPU hour and make it feasible to run such systems without the need of a GPU.
2. **MEOw experiments and results**: The approaches were evaluated on a popular large dataset - SIXray (Miao et al. 2019) and a proprietary dataset called "Residuals" to assess the feasibility of the presented approaches for concealed threat identification (Section 5.1.4). For SIXray, MEOw yielded promising results by dis-

⁴Adapt the newly developed method to real-world dataset using concealed threat detection problem as a case study to showcase potential field applications. .

⁵Develop a heuristic approach to bolster the feasibility and efficiency of AutoML in both model and data optimisation, facilitating its broader adoption in industrial applications, exemplified through a case study in concealed threat detection.

covering an architecture that achieves 87.87% mAP, which is 10% higher than the next best state-of-the-art architecture- ResNet50 (He et al. 2015). Testing MEOw on the commercial proprietary dataset also results in success, but here the MEOw-produced models achieve 2%-5% better results, up to 91.23% accuracy for predicting what the threat is and up to 89% accuracy as to what type of modification is present. This comparison might not look convincing enough, but since task error is only one of the objectives, the models produced by MEOw use less than half of the second-best architecture and when compared to similarly sized models the difference increase from 2%-5% to more than 20%. All in all, state-of-the-art results are produced on large-scale X-ray concealed weapon dataset known as SIXray using RAMOSS, which indicates that the rationale behind the approach is sound, and it is in line with **Objective 4**⁴

3. **Segmentation Threat identification:** An additional experiment is conducted to establish the feasibility of RAMOSS for segmentation threat identification with an extension of the Residuals dataset. Table 5.4 revealed that, once again, the methods presented throughout the thesis achieve an optimal balance of objectives. The RAMOSS-produced architecture achieves 2%-4% worse test f1 and IOU scores, but the architecture is composed of 60-100+ times fewer parameters. Compared to similarly sized (yet still more than 10 times larger) architectures (He et al. 2015, Szegedy et al. 2015b), it achieves over 20% better results on the test set. Interestingly, during the validation, the RAMOSS architecture achieves the worst results, and it is the only architecture to display less than a 1% difference between the validation and test results, making it superior in terms of generalisation error.
4. **MIRA-ME: Coreset discovery method:** Based on the modularised design of RAMOSS, this chapter demonstrates that the algorithm can be repurposed for coreset discovery. Both MIRA-ME pipelines possess a unique ability because the images are encoded continuously, allowing for a better understanding of the "importance" of each sample of the dataset. Both pipelines of MIRA-ME automatically determine the needed coreset size in difference to the state-of-the-art (Ju et al. 2022, Killamsetty et al. 2021) where it needs to be specified explicitly before the search. In contrast to previous work (e.g. Barbiero et al. (2019 2020)), MIRA-ME provides an extra layer of interpretability since the encodings of the coresets can be explored more closely and serve as a proxy for estimating individual image "importance". It also allows for user preference articulation by adjusting the threshold for selecting instances, which enables the control of how many samples are to be chosen post-process. Through adjusting the reference points for the different objectives, a prior can be introduced that biases MIRA-ME to weight objectives differently and produce even more customised results.

5. **Coreset results:** Some interesting findings unrelated to MIRA-ME are discovered during the experimentation. State-of-the-art approaches (Killamsetty et al. 2021, Ju et al. 2022) struggle to operate in the low-data regime, while studies aim to go down to 30% or 10% of the dataset in the MIRA-ME experiments, some of the methods display worse performance than Random when they are tasked to search sub-10%. For example, for MNIST, Glister at 1% is outperformed by Random with more than 20%; in fact, Random outperforms Glister at 1% when it uses only 0.1%. MIRA-ME, on the other hand, achieves excellent results in both sub-10% and sub-1%. MIRA-ME outperforms EvoCore when 0.1% of MNIST(60-82 images) is used and significantly beats Random. A general tendency can be observed that the more complex the dataset becomes, the more significant the gap between MIRA-ME and the rest of the approaches. While for MNIST (97.67% and 95.7% with 4918 and 1859 samples for the "naive" and the "unsupervised" pipeline respectively versus 98.5% for Glister with 6000 samples), the benefits of MIRA-ME are only apparent when compared to Random sampling(95.68% with 6000 samples), in CIFAR-10(63.5% with 4500 samples for the MIRA-ME pipelines versus 40% and 53% for SimCLR and Glister respectively with 5000 samples) and especially in SIXray, the trend becomes indisputable. For SIXray, the MIRA-ME "unsupervised" achieves 62% with only 3536 samples compared to 38.6% for Glister with 7496 samples.

Next, some implications of the project are presented:

6.2 Implications

1. **Automatic Discovery of Optimal Convolutional Autoencoders.** Convolutional autoencoders produced by MONCAE can be used for data exploration, filtering and rapid prototyping. They are powerful tools, and their bottleneck representations can be treated as a proxy for the original input. Because of their reduced size and theoretically higher information density, bottleneck representations can utilise them to explore and visualise vision data more efficiently, filter out conceptual outliers, and make possible the separation of feature extraction and feature selection that can accelerate model/hyperparameter and other time-consuming searches. Although this thesis does not fully explore how such encodings can be applied, it is a promising direction for future work. Moreover, based on the experiments conducted in Chapter 3, reconstruction loss might not serve as a good enough proxy for the quality of the compressed representations. Thus, there is a need for new methods of evaluating these bottleneck representations and of ways to leverage their explainability potential optimally. More broadly, identifying how specifically models encode "concepts" in these bottleneck layers could render in the future potential ground-breaking dis-

coveries in Explainable Artificial Intelligence (XAI) (Kazhdan et al. 2020).

2. **Streamlining pre-processing.** Combining the discovered convolutional autoencoders with the field of XAI (Dimanov 2021) can yield excellent results and enable a quicker data cleanup and label verification (Kakani et al. 2020), as suggested from the motivation experiments in Chapter 3.
3. **Fostering transformers efficiency in computer vision.** Recently, there has been an increased interest in using transformers for addressing computer vision tasks, originally designed for sequence modelling. The application of transformers, however, requires a substantial amount of heuristics and their use is still hindered by some of the same caveats as AutoML. Thus, a good future work direction would be to explore the use of neuroevolution for speeding up this time-consuming process by designing more optimal representations and attention mechanisms with the help of neuroevolution that can ultimately speed up transformers, improve their training process to resemble the high dimensional agreement present in capsule networks (Patrick et al. 2022, Sabour et al. 2017). Especially the contribution of this thesis in Chapter 3 gives insights into how neuroevolution can be used and guided to generate good-quality representations.
4. **Substantially reducing computational costs.** The computational time saved by employing the presented methodologies would allow for the fine-grained customisation of produced solutions and the more efficient use of researchers' and practitioners' time (Valvano et al. 2018). In addition, one of the largest time sinks in the business world is precisely this wait for the model to be discovered and trained⁶ which often impedes the ability of deep learning experts to proceed with further stages of development.
5. **Customised dataset compression.** Since MIRA-ME produces multiple coresets, a strategy similar to the one performed with the ensemble method in Chapter 5 can utilise all different weights for the instances. This strategy would provide a user-adjustable threshold for controlling the coreset size (the degree of compression). The potential incorporation of Progressive Stratified Sampling (PSS) with subsets produced by MIRA-ME and downsampling even further by a specified coefficient will take this even further.
6. **Offsetting the carbon footprint of deep learning approaches:** Last but not least, a growing concern in the machine learning community is the environmental impact, and carbon footprint of different methods Lacoste et al. (2019), to the extent that authors in many prestigious conferences (e.g., ICLR, ICML, NeurIPS, AISTATS) are

⁶Based on personal experience and informal interviews with industry partners.

required to submit their estimated GPU hours as well as carbon impact (calculated with tools like MLCO -<https://mlco2.github.io/impact/>). One of the most significant contributions of this thesis is accelerating AutoML and NAS. Consequently, this thesis contributes to a greener future by progressively lowering the immense carbon footprint of these approaches (Dimanov et al. 2022). The results presented in Chapter 5.1 suggest that heuristic methods may be effective in exponentially accelerating AutoML approaches and, more broadly, automatic decision-making. These findings highlight the potential benefits of using heuristic methods and the importance of an engineering mindset in addressing the computational challenges associated with AutoML. Future research should investigate the generalisability and limitations of these approaches to understand their potential impact on the field entirely.

6.3 Future work and limitations

Even though this thesis addresses big research questions in AutoML and potential industrial applications in security, the findings and wider implications of this work generate an exorbitant amount of new intriguing research questions. Next, some of the future research directions that has been deemed most promising and important is listed:

1. **Proxy scores:** NASWOT (Mellor et al. 2021) and SYNFLOW (Tanaka et al. 2020) can accelerate AutoML and NAS by exponentially decreasing the needed time for training architectures upfront. However, these proxies usually use just a single batch of data, which, more often than not, is not representative of the whole dataset that they are applied to. In the last chapter, this thesis explores how datasets can be compressed by several orders of magnitude, allowing for the discovery of critical datasets. Furthermore, some other methods that offer synthetic generation of samples offer to compress datasets to a single sample per class (Wang et al. 2018).

In their current shape, the methods from this work cannot be feasibly applied in general AI scenarios, as discussed in Chapter 5. However, datasets can be substantially compressed with the advancements proposed by such methods and the data reduction approaches presented in MIRA-ME (Chapter 5) and PSS (Chapter 4). Hence highly complex models can be potentially iteratively evaluated on a reduced number of batches by using proxy scores introduced in Section 5.1.2. Thus, the combination of the new approaches might prove to be a closed-loop system capable of not only assisting AutoML systems but also aiding in the general training cycles of machine learning algorithms and, more importantly, in compiling better data understanding techniques.

2. **Proxy scores, MEOW and MIRA-ME:** More specifically, a possible application of

proxy scores can be achieved using MEOW's acceleration and MIRA-ME's compression. This way, more accurate proxies can be constructed, especially if the Progressive Stratified Sampling implemented in RAMOSS is used on top of the discovered MIRA-ME coreset together with the heuristic speedups of MEOW. For example, the NASWOT score (Mellor et al. 2021), as well as SYNFLOW (Tanaka et al. 2020), use a batch of data to determine the score of the network, but what if this batch is indeed the maximum compressed coreset of a dataset and if it is not small enough then it can be iterated using PSS.

3. **HPO: Hyperparameter Optimisation:** This thesis provides several different methods of achieving NAS. However, the thesis does not extensively explore all the different effects of hyperparameter tuning for the neuroevolution controller algorithms, which would greatly benefit the interpretability of the present techniques. Moreover, a standard framework should be composed (similar to CME (Kazhdan et al. 2020) and (Kazhdan et al. 2021)) that objectively evaluates the interpretability of AutoML methods. This framework can be added as a separate objective in any of the proposed methods, as they are designed with scalability in mind. Furthermore, the effect of the interpretability of using multiple different produced architectures from the set of solutions should also be the focus of future work.
4. **Exploring the final population:** Evolutionary algorithms used throughout this thesis generate an approximation set of solutions. One way this is utilised is by combining them via an ensemble approach in 5 to generate an optimal solution. Based on the successful results, future work should explore other ways of exploiting the generated approximation set of solutions.

The use case with the ensemble approach merely suggests the potential of combining some of the produced solutions. However, future work should focus on further optimising this process and extracting correlated features (possibly from the different genomes), possibly summarising the solutions into a single one through some sophisticated aggregation procedure (e.g., (Abdelfattah et al. 2021) subject to preference articulation and allowing further customisation.

5. **Use of MONCAE as part of the embedding and linear projection in Transformers:** With the current rise of transformers, researchers are looking for more ways to apply them to various problems. Yet, there are only a few standard transformers currently used. Future work should explore the use of MONCAE (chapter 3) as means of discovering convolutional autoencoders that can then be integrated as part of the transformers. Moreover, future work should explore using autoencoders and variational autoencoders as potential substitutes for embedding and linear projection.

6. **Neuroevolution for Capsule Networks:** To conclude, there might be potential for using the proposed approach with capsule networks and discovering capsule networks instead of CNNs. Capsule networks are currently underutilised (Hinton et al. 2018). However, researchers may make better use of their equivariant representations in the future, and all novelties in MONCAE, RAMOSS, MEOW and MIRA-ME are built in a modularised way allowing for the swap of convolutions for capsules. Modifications of the training during the evaluation process might be needed given the specific dynamic routing used by capsules (Sabour et al. 2017). However, the high-dimensional agreement between agents in these approaches may resonate with the idea of producing an approximation set of solutions and exploring the per-generation intra-population high-dimensional agreement between different phenotypes.

This thesis presents a suite of novel approaches for automated machine learning in computer vision. These approaches are specifically designed to foster the efficiency of current AutoML methods in various fundamental computer vision problems by addressing some pressing challenges, such as their high computational cost. Results show the (still not fully unleashed) potential of multi-objective and heuristic-based techniques in a wide range of critical applications, such as threat detection. Hopefully, the findings from this thesis will contribute to the wider adoption of automated machine learning in computer vision and lay the foundation for future research in this rapidly evolving field.

Appendices

Appendix A

X-Ray screening

X-Ray screening is used in a wide range of settings, and hence is currently playing a crucial part for e.g., enforcing security in publicly crowded areas and in medical applications . Both scenarios require experts with a specific skill-set to be able to examine the produced images, which introduces multiple different possible point of failure, which can be caused by a variety of factors, such as exhaustion and distraction (Liang et al. 2019b) . More specifically, in the context of threat identification, airport bag screening can be used as an representative example. In this use case, the job of the transport security officer(TSO) involves looking at many different X-ray images, identifying potentially hundreds of different threats accurately, and also achieving this in a timely manner, since throughput of passengers in airports is also an important requirement to balance (Liang et al. 2019b). Given these circumstances, in the recent years the industry has started making efforts to implement an automated approach to aid manual operators (Department of Homeland Security 2017)(Liang et al. 2019b)(Chavaillaz et al. 2019).

Discovered in 1895 (Röntgen 1895) by Wilhelm Röntgen, the ability of X-rays to penetrate objects and identify inner structures found use in many different settings and industries including medicine, security, archaeology and many others (Mery 2015)(Papadopoulou et al. 2007). Through out the years, X-rays have played a fundamental role in everyday life applications ranging from routine medical check-ups and non-invasive monitoring for fractures or medical conditions and diseases to baggage scanning in airports (Mery 2015).

As shown in Michel et al. (2007) there was a substantial increase of operators' performance in x-ray image interpretation resulting from adaptive computer-based training, which connotes the importance of assistive computer systems for threat detection. Furthermore, the same study discovered that objects were far more missclassified by human operators when they were rotated in an unconventional way as well as that "screeners could reduce the time needed to detect a threat object significantly" (Michel et al. 2007). A byproduct of the research recognised by Mery (2015) was also that the human detection performance during peak hours is only about 80-90%.

After 9/11 a new urgent need for improved and advanced threat detection started to emerge (Mery 2015). In fact, back in the late 80's Murphy (1989) the state-of-the-art detection systems for 1989 already included Thermal-Neutron Activation, Fast-Neutron activation and also dual-energy X-ray systems, which later lead to the development of 3D computer tomography using X-rays for a more accurate detection of explosives (Mery 2015).

Currently, most airport systems use two 2D X-ray views (top-down and sideways) to complete baggage screening (Liang et al. 2019b) , but recently new and more advanced computer-topography (CT) based systems have been proposed and implemented in some (Gaus et al. 2019) (Akçay and Breckon 2020). The CT scans differ from the conventional and traditionally performed X-ray scanners by having the ability to display objects in 3D rotatable plain, thus giving operators more rich information about the scans (Mouton and Breckon 2015). However, the automated analysis of 3D CT scans is much more complicated not only because of the added dimension, but also the intrinsic noise and artefacts present and also the increased demand for computational power (Mouton and Breckon 2015).

In short, X-ray scanning is a huge part of concealed threat detection and it is so widely used in security on a global scale . Thus, a major part of the research in this Thesis will focus on analysis and concealed threat detection in X-ray images.

Appendix B

Convolutional Neural Networks

As evident from the discussion in Section 2.2, the wide use both in industry and research of convolutional neural networks has led to astonishing progress. This section presents a brief explanation of how these networks operate, which is integral for understanding of the rest of the thesis.

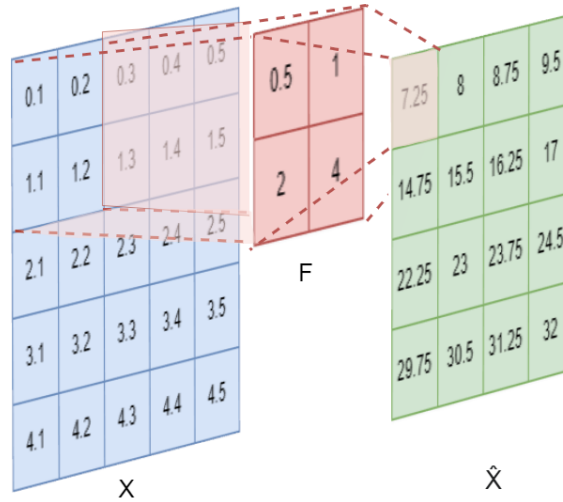
Convolutional neural networks(CNNs) are artificial neural networks which use specialised layers (convolutional and pooling layers) to deal with high dimensional and complex inputs (Wu 2017).

Ever since LeCun et al. (1998) used convolutional neural networks to achieve state-of-the-art performance in computer vision, CNNs have been one of the most widely spread machine learning techniques for image processing, image classification, feature detection, feature extraction and much more (Goodfellow et al. 2016b). The unique properties of CNNs and the fact that convolutional as well as pooling layers use kernels instead of fully dense connections, allows them to be especially good at learning representations with multiple different levels of abstraction (Yang et al. 2017a) and effectively process high-dimensional and complex data (Pereira et al. 2009).

B.1 Convolutional layer

Convolutional layers produce feature maps, which are different representations of their input (Yoo et al. 2015). They achieve this through the application of trainable filters which produce a linear map of the input as displayed in Figure B.1. The number of these feature maps and filters is a hyperparameter, which is fine-tunable based on the particular application of CNNs at hand. Typically, the tuning process starts with an initial number of filters, which is then increased in a reversely proportional fashion to the compression of the input image through pooling layers or strided convolutions (Liu et al. 2019a). Through these various linear transformations and combining them with non-linear functions such as the rectified linear unit, CNNs are adept at handling high dimensional inputs (Wu

Figure B.1: An example convolution operation with a kernel (F) of size 2×2 (m, n) and a stride of $1, 1$ (s_x, s_y) iterating over 5 by 5 input (X) and producing 4 by 4 feature map (\hat{X}) where each $\hat{X}_{i,j} = \sum_{h=0}^{s_x-1} \sum_{w=0}^{s_y-1} F_{h,w} X_{i*s_x+h, j*s_y+w}$. So when the stride is $1, 1$ it can be simplified to $\hat{X}_{i,j} = \sum_{h=0}^{s_x-1} \sum_{w=0}^{s_y-1} F_{h,w} X_{i+h, j+w}$.



2017). Towards this goal, convolutional layers use kernels of predefined size and strides, which iterate over the image. These kernels can also be of different shapes (spatial size), which is one of the factors of variation which is considered when designing an attention mechanism for CNNs or just generally when dealing with convolutional neural networks (Gu et al. 2018). The kernels store all trainable parameters of the different filters in a particular convolutional layer.

The purpose of the convolutional layers is to make the features in a given image more prominent and to conduct feature extraction in general (Albawi et al. 2017). Thanks to the convolutional layers, certain features are extracted, which can then be used to solve a given problem. There are studies which theorise that good networks should produce explainable representations which inherently contain "concepts" (Kazhdan et al. 2020). This is discussed in further detail in Appendix C. Using these concepts, convolutional layers can be directed to look for certain features to extract instead of relying purely on the loss minimisation gradients flowing through the classification layers.

Because convolutional layers naturally shrink the dimensions of the input image along the edges based on the movement of the kernel displayed in Figure B.1, some of the spatial information in these regions may be lost. Hence, a technique called padding, is used to ensure these values are not lost when the input size is reduced (Albawi et al. 2017).

There are several different popular padding techniques, of which the most popular ones are zero-padding, same padding and valid padding (Wu 2017).

Another interesting hyperparameter of convolutional layers is the stride. The stride of a convolutional layer determines how big the step the kernel takes (pixel-wise) when

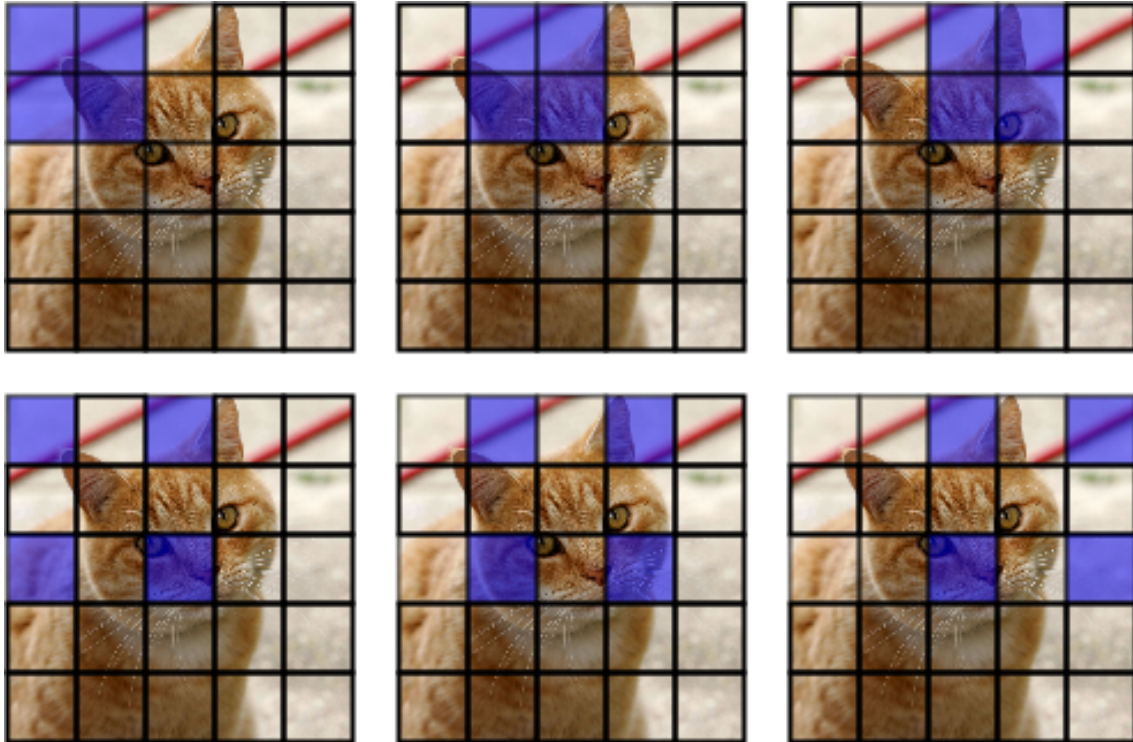


Figure B.2: Dilated convolution kernel compared to a normal convolutional layer kernel

transversing through its input. Usually, the stride is set to $(1,1)$, which means that the kernel moves 1 pixel when going from left to right and from top to bottom, but it can be modified to be more than 1 in which case the size of the produced feature map will be smaller than the input in proportion to the stride step amount.

New types of convolutions have also emerged. Here, one of them (dilated convolution) is covered. It can be important for understanding some of the work in Chapter 4 since dilated convolutions are put as possible layer types in the search space of RAMOSS. The reason for considering dilated convolutions in this work is their effectiveness in segmentation problems (Hamaguchi et al. 2018). Discussion on transpose convolution can be found in Section 2.1.

The dilated convolution (often also called "atrous convolution") works in the same way the normal convolutional layer does, but instead of the kernel being a densely connected block, the parameters of the kernels are dilated (as depicted in Figure B.2 (Wu et al. 2019a)). This introduces one more hyperparameter called dilation rate (Lei et al. 2019) , which is responsible for controlling the separation of parameters between different parameters of the kernel. For example, an atrous 5 by 5 kernel with a dilation rate of 2 will have the same receptive field as a 9 by 9 kernel, but only 25 parameters instead of 81. They are usually used when a wider field of view is required but there are computational limitations which make it unfeasible to add more layers or expand the model's number of parameters.

B.2 Pooling layer

The pooling layer aims to reduce the size of the feature maps and in theory, select the most important information in the feature maps and discard the less significant information. The purpose of the pooling layers is to compress the data representation to lower-dimensional vectors (Sun et al. 2017) while conserving a high level of information.

They also use a kernel with a specific stride and size, but this time instead of using a kernel with trainable parameters, the kernel method is simply assigning a summary value (like the average or the maximum value in the kernel). This value is then what ends up in the feature map (Scherer et al. 2010).

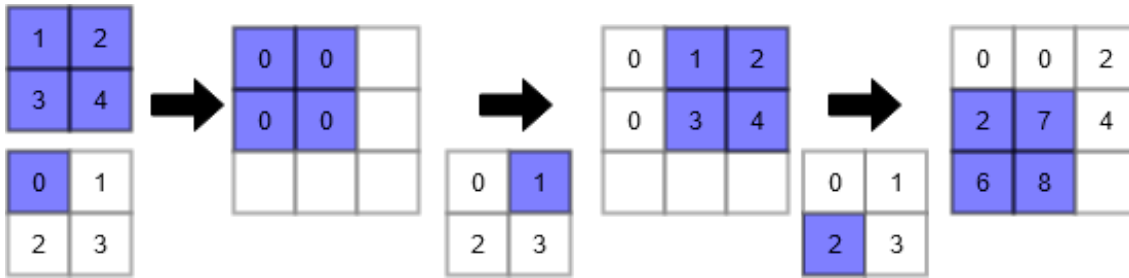
In short, the pooling layer aims to conduct subsampling over the feature maps produced by the convolutional layers to decrease the spatial size of the maps (Sun et al. 2017). It is one of the quintessential operations in convolutional neural networks together with the convolutional layer itself. However, some work explore using strided convolutional layers instead of pooling layers since instead of simply applying a predefined reduction operation, one can use the trainable parameters that can extract information out of the feature maps more efficiently. The way strided convolutions work is that the kernels move at a different pace rather than the default 1x1 stride described above. This way, the layer can achieve the same compression as a pooling layer without sacrificing the trainable parameters.

B.3 Upsampling layer

The upsampling layer aims to reverse the operation of the pooling layer (Goodfellow et al. 2016a). This is usually done to restore the original dimension of the image which is an integral part in convolutional autoencoders and segmentation models (Ronneberger et al. 2015a). The upsampling layer uses a kernel similar to the pooling layer, but instead of mapping to a lower-dimensional representation space, it maps to a higher-dimensional one. There are two general ways to achieve this in a convolutional neural network. The first one is to use an upsampling layer, which scales up a data representation using a nearest neighbour or bilinear upsampling.

The second one is to use a transposed convolution, the inverse operation of a strided convolution. Transposed convolutions came to light as a means to upsample features using a kernel with weights (Zeiler and Fergus 2014). These are also termed 'deconvolutions' or 'fractionally strided convolutions'. An example of a transposed convolution is presented in Figure B.3. The figure depicts the iterative process of the kernel (top left 2x2 grid) going over the input (the bottom 2x2 grid). Each cell from the input is multiplied by the kernel at each step. Then, all produced feature maps are added using a specified step size (stride). In the example, the stride is 1, so each cell where there is overlap

Figure B.3: The process of upsampling with a transposed convolution with a 2x2 kernel and a stride of 1.



represents the sum of all produced overlapping cells.

B.4 Hierarchical Convolutional Neural Networks

According to Seo and Shin (2019) one of the biggest challenges for CNNs due to their discriminative nature Dai et al. (2014) arises when two or more classes share visual similarities (Agyemang and Bader 2019). Seo and Shin (2019) explain that Hierarchical Convolutional Neural Networks (H-CNNs) tackle this issue by using a two-step process, which resembles a sieve. First, the classes, which have fewer visual similarities are processed and then a second step deals with classes that look alike (Seo and Shin 2019).

H-CNNs also provide the opportunity to add additional classes to previously trained, well-performing domain-specific classifiers. This is implemented to prevent the caveat associated with CNNs that they suffer from the inability to classify previously unseen entities, which do not belong to any of the classes seen during training (Agyemang and Bader 2019). Indeed, it has been proposed that a future use-case for H-CNN could be for anomaly detection” (Agyemang and Bader 2019).

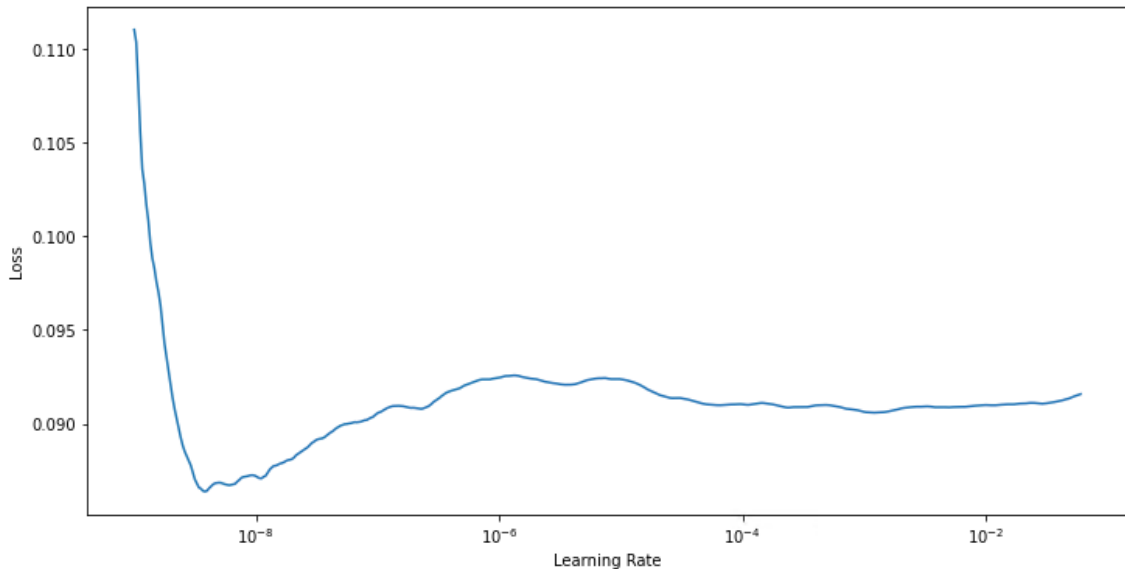
On the other hand, a disadvantage of H-CNNs is their complexity and increased training time as they have to “train each CNN within the hierarchy while also having to fine-tune the hyperparameters for each CNN architecture such as deciding the number of convolution filters” (Agyemang and Bader 2019).

B.4.1 Hyperparameters

Training a deep neural network to learn to solve these problems is not a trivial task (Goodfellow et al. 2016b). So far, we’ve discussed picking the correct architecture and optimisation methods, but determining hyperparameters, how to choose them, and how to adjust them, is also vital (He et al. 2021). This section examines some of the most imperative hyperparameters for creating training/testing neural networks pipelines, their importance, and immense impact on the performance of different learning algorithms.

Learning rate and momentum: Arguably the most important hyperparameter when

Figure B.4: Figure depicting the result of a learning rate finder. The learning rate is plotted together with the corresponding loss throughout 1 epoch of learning rate change. Learning rate is changed every n^{th} training step, where n is subject to the number of learning rates that will be explored and the number of mini-batches of the data. The steepest fall signifies a good initial learning rate.



training a neural network is the learning rate (Sutskever et al. 2013) . The learning rate determines the magnitude of the weight adjustment the optimiser would perform at each step that it takes (Darken et al. 1992) . It is of extreme importance since a too large learning rate can result in skipping over optimal regions of the loss function and only superficially exploring the loss space, but also too small learning rate can easily lead to the optimiser getting trapped in a local minimum or saddle point (Goodfellow et al. 2016b). An optimal learning rate would lead to the greatest step towards gradient improvement, which can be found rather simple by allowing the algorithm to train for an epoch with a set of different learning rates and determine the largest gradient (Figure. B.4), serving as a starting point for the training of the algorithm. This process can theoretically be done at every training step to achieve optimal performance, but it is often not practical for computational cost reasons. Thus, some common practices include decaying or annealing the learning rate with the progression of the training (Shen et al. 2022) or recently the successful application of "one-cycle" learning. It is a two-step process that first starts by increasing the learning rate gradually followed by a drastic decrease, which has proven to be an effective way of training deep neural networks for a variety of problems (Smith 2017).

Number of channels in each layer: Another vital hyperparameter, which is part of the architecture choice is the number of filters or nodes in each layer. The process of convolutional filters work has been discussed in Section B and nodes in fully connected

layers, but choosing how many of these filters or nodes is often a manual process, based on good practices and experience which have worked well in some general benchmarks (Ahmed et al. 2020) . An example of this strategy consist of using a smaller number of filters in the first convolutional layers and then increase their number inversely proportional to their size (Liu et al. 2019a). In essence, when the pooling is completed (regardless if it is done with a dedicated pooling layer or strided convolution) the height and width of the convolutional filters are halved, and the common practice is at this point to double the number of filters. The reasoning behind this good practice is to minimise the loss of important information due to the spatial compression of the input (Bender et al. 2020). The starting point of the number of filters varies greatly across literature but some good examples are 32 or 64 filters (Liu et al. 2019a, Ronneberger et al. 2015b). Although this hyperparameter depends strongly on the dataset at hand, in difference to the methods for finding out a good learning rate there are no real well-established methods for determining what a good starting number of filters is. An alternative, to the otherwise trial and error process, is the field of explainable AI. Explainable AI deals with the robustness and interpretability of neural networks and is concerned with questions exactly like this (Frankle and Carbin 2018, Kazhdan et al. 2020) . Informally, these approaches attempt to categorize the different knobs of the black-box optimisation control room of a neural network (Dimanov 2021). They use implicit properties of the data itself to determine the need for as many of the parameters and hyperparameters as possible.

Batch size:Ideally, neural networks should backpropagate the gradients from the whole dataset that is fed into them, but with large-scale datasets, this is computationally infeasible. Hence, usually the dataset is split into mini-batches (Hinton 2012) and fed to the algorithm in smaller chunks. There are generally two approaches to working with batches and one is to evaluate all mini-batches and update the networks once all of them are evaluated (Ruder 2016) or to evaluate the mini-batches one by one and update the gradients and weights at each of this mini-batch steps (Khairat et al. 2017). Of the two, the second approach is far more popular.

The size of these mini-batches ("batch size") is one of the easier parameters to experiment with because of the numerous studies agreeing that up to a certain point (varying between 256 and 512) it is generally good to opt-in for the highest possible batch size the hardware limitations allow for (Radiuk et al. 2017, Neishi et al. 2017, Smith et al. 2017) .

Appendix C

Representation Learning

Representation learning is a flourishing field which is integral to the design of interpretable models (Dimanov 2021), that is often overlooked in deep learning. According to Marr (2010), it is the process that "makes explicit certain entities and types of information" and attempts to achieve some information processing goal.

The ability of humans to solve mathematical problems is hindered by their understanding of what the different digits, symbols and operations represent as evident from the difference in performance of humans to perform complex calculations with Arabic versus Roman numerals (Marr 2010). The same applies to deep learning systems. For a deep learning model to have a good performance, it needs to restructure the input data into fitting representations of this data (Alqahtani et al. 2021) . The whole purpose of the convolutional and pooling layers in convolutional neural networks is exactly this (Goodfellow et al. 2016b). But after industry moved from HAAR and similar features to convolutional neural networks and other deep learning models, the construction of these representations has been made more and more black-box optimisation and concealed behind other processes and layers (LeCun et al. 1998) . Representation learning aims to unveil this essential part of machine learning and gain more control and understanding of what makes a good representation and how to steer a model to achieve it (Dimanov 2021). The work of Dimanov (2021) for example explores the different assumptions and priors of representation learning and discusses how to qualify a good representation using information about its features and factors of variation. This is an important stepping stone in explainable AI and deep learning in general since in contrast to supervised learning and other methodologies, which have well-defined objectives, representation doesn't (Dimanov 2021). Being able to score representations can thus be used to formalise a general objective function, which can be used to train or/and improve representations. An example of such an objective function for representation is the work of Bengio et al. (2013), which describes the ideal representation as being able to "disentangle as many factors as possible, discarding as little information about the data as practical". This begs the question of the limits and how much data loss is "practical" and also how much is

”possible” (Dimanov 2021, Goodfellow et al. 2016a).

This quality of representations is often referred to as the ”representational power” or ”representational capacity” (Arpit et al. 2017) and there have been multiple recent attempts to boost the representational capacity of various neural networks (Turner et al. 2021) . There are also findings which suggest that the deeper neural networks can produce more concise representations which still conserve all needed information by employing the use of generated concepts inside the hidden layers of the networks (Bermeitinger et al. 2019). An interesting finding of the relationship between the growth of a neural network and its representational capacity presented by Bengio et al. (2003) states shows that shallow networks grow exponentially the more complex the representations get, whereas deeper networks grow polynomially. However, in their study (Bermeitinger et al. 2019)’s empirical results suggest that shallow networks might have an edge over deep networks in ”attaining low mean square minima for given mapping problem”. Different optimization algorithms result in vastly different gaps between deep and shallow networks, suggesting that the study’s findings might not be in direct opposition to the widely held idea that more layers lead to better performance (Bermeitinger et al. 2019). This instead shows that training a model with a higher representational capacity requires better optimization and longer time to converge.

Appendix D

History of computer vision

One of the first projects in this field was "The Summer Vision Project" (Papert 1966), which attempted to use a computer and a camera to achieve pattern recognition by applying heuristic rules. Many image classification algorithms were then researched using different approaches (Haralick et al. 1973, Szummer and Picard 1998, Chapelle et al. 1999). One of the first object detection algorithms, which were efficient and worked on real-time inputs was the Viola-Jones algorithm (Viola and Jones 2004). It used HAAR-like encoded features and then compared the gradients of the pixels in the images to these features with the use of a Support Vector Machine (SVM), thus achieving face detection.

Later, different methods (Dalal and Triggs 2005), feature descriptors and similar approaches were introduced, but the next major advancement in the field came when a technology, dating back to 1957 (Rosenblatt 1957), was reintroduced in 2012 by the team of Krizhevsky et al. (2012) under the supervision of a respected figure in the field of artificial intelligence- Geoffrey Hinton. The new approach was to use a specific type of neural network- the convolutional neural network (CNN), which was proposed by a team of researchers again supervised by Hinton in a paper (Lang et al. 1990) and further researched and developed by LeCun et al. (1998), in an attempt to achieve state-of-the-art performance on the largest publicly available image dataset- ImageNet¹. The newly designed AlexNet (Krizhevsky et al. 2012) outperformed all previous models used in the annual competition ImageNet LSCRV². This new revolutionary, yet old, idea was able to work much better than its predecessor, because of the vast increase both in terms of computational power and data available in 2012 compared to the late 1960s (Alom et al. 2018).

After the 'rebirth' of the idea to use convolutional neural networks for computer vision many different variations and architectures started to dominate other models. The accuracy for public datasets and in different competitions started to rise almost exponentially. Big companies like Google and Microsoft started to get involved in the field and in the

¹<http://www.image-net.org/>

²<http://www.image-net.org/challenges/LSVRC/>

following years models like VGG (Simonyan and Zisserman 2014), ResNet (Targ et al. 2016), Inception (Szegedy et al. 2015a) and many others started to emerge. By 2015 researchers found that computers were outperforming humans for some visual tasks (Wu et al. 2015).

Many different variations of the models were then proposed and with the advancements in computational power, the models started scaling up and becoming more and more complex (Liu and Deng 2015, Xia et al. 2017, Szegedy et al. 2017a). However, a major deficiency started to arise. The CNNs were designed to solve image classification problems (one image represents one entity or object), but an object detection was needed to mimic the visual ability of humans (one image can contain multiple different objects) (Girshick et al. 2014).

Thus, the idea of combining region proposals with CNNs was proposed- resulting in R-CNN (Girshick et al. 2014). This specific branch of CNNs focussed on solving the object detection and semantic segmentation performance stagnation (Girshick et al. 2014). The idea behind this new approach, which achieved state-of-the-art performance, for the time it was proposed, is that it uses an algorithm to determine the positions of potential objects and then employs the ability of CNNs to classify images in these regions. Later, many different variations of this approach started to appear with Fast R-CNN (Girshick 2015), Faster R-CNN (Ren et al. 2015), Mask R-CNN (He et al. 2017) and others, which also achieved state-of-the-art or near state-of-the-art performances when they were released and benchmarked (Jiang et al. 2018). In the same year that R-CNN was proposed another revolutionary idea on generative modelling, the General Adversarial Networks (Goodfellow et al. 2014) Goodfellow et al. (2014) are based on game theory. and in essence two separate neural networks are cooperating and competing in the same time to produce a good enough agent that generates good "fakes" and a good "detective" that can discriminate fakes and real samples.

The next big innovation in the field of computer vision, which achieved state-of-the-art performance on the MNIST dataset ³ was the proposal of Capsule Networks (Sabour et al. 2017) and the corresponding EM-Routing algorithm (Hinton et al. 2018), which used a revolutionary approach and proposed adding a whole new structural element to the neural networks. With Capsule Networks the team of Hinton et al. (2018) identified a deficiency with previous computer vision approaches. Firstly, convolutional neural networks suffered from their integral simplicity since they contained just a few levels of structure. In Capsule Networks a new level of structure is introduced beyond the layer. This new building block of a neural network, the **capsule**. was inspired by the mini-columns present in human brains and associated with human vision (Buxhoeveden and Casanova 2002). Capsules also contain neurons which are stacked vertically. The capsule is designed to store different properties of an entity and is composed of multiple neurons, which cor-

³<http://yann.lecun.com/exdb/mnist/>

respond to a property of this entity (ex.: position, orientation, size, colour, intensity). At a high level, the capsules combine multi-dimensional vectors, which represent an entity and its **pose** (position, orientation and other properties) and output a probability of the presence of the entity and its pose via "routing by agreement"(Hinton et al. 2018). This operation adopts the idea that if high-dimensional entities agree on every dimension that a pose is important within a given small variation, then the chance of this happening is significantly low, and the opposite if multiple entities agree sharply, then a certain entity should be present in a certain pose(Hinton 2014).

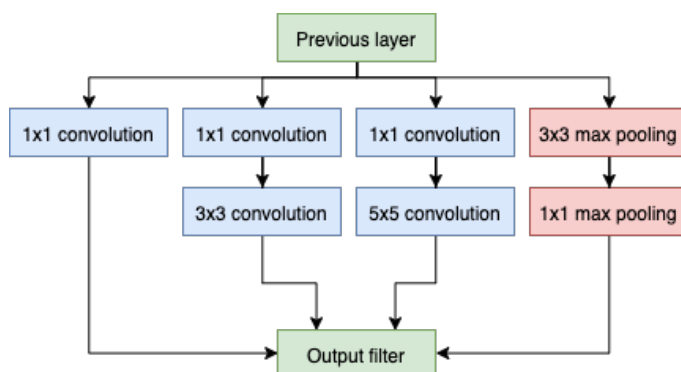
The latest advancements in the state-of-the-art in computer vision are a product of the automated machine learning (AutoML) efforts of researchers and industry experts, who have attempted to design systems, which are capable of automatically discovering a way of solving a given problem. In the domain of neural computation, AutoML is a process of discovering the right hyperparameters(layers, number of neurons per layer, order of layers, types of layers etc.) and parameters (weights and biases) of a neural network (Hutter et al. 2019b). A whole new branch of algorithms has emerged, which deals with neural architecture search(NAS) (Elsken et al. 2018b). There are many different types of algorithms used for neural architecture search, but the two most popular ones are reinforcement learning and neuroevolution (Stanley et al. 2019a).

Appendix E

State-of-the-Art Computer Vision Methods

InceptionNet: In 2014 researchers at Google proposed a new architecture called Inception (Szegedy et al. 2015a), which achieved state-of-the-art results on the popular ILSVRC 2014 benchmark. A specific instance of the architecture is referred to as GoogLeNet, which is a 22 layer InceptionNet used in ILSVRC 2014. It was the winner of ILSVRC 2014 for "object detection with additional training data" both in terms of categories found and mean average precision. The main aim of InceptionNet was to optimise the width and depth scaling while keeping the computational cost as low as possible (Szegedy et al. 2015a). The architecture was designed to allow for both depth and width scaling and one of the main advantages it has over previously designed convolutional neural networks is the degree of spatial exploration. It was intended to optimise the process of neural architecture construction (Szegedy et al. 2015a). This was also one of the first times the idea of convolutional blocks was proposed.

Figure E.1: Inception block used in Szegedy et al. (2015a).



InceptionNet is based on the Inception block (Figure E.1), which contains multiple convolutional and pooling layers embedded inside it. The first version constricted the filter sizes used in this block to just 1x1, 3x3 and 5x5 ones mainly to avoid patch alignment.

First, a different Inception block was created, but changed after considering computational performance. Dimension reduction was added in the form of a 1x1 convolutional layers before the 3x3 and 5x5 ones, which is done for to conserve resources and because of the use of ReLU activation inside. Additionally, the Inception block is created to facilitate stacking multiple blocks one after another. Thus, the output of every block is designed such that it can be used directly as an input for the next Inception block (Szegedy et al. 2015a).

In GoogleLeNet the network is constructed by substituting some of the fully connected layers by average pooling. Also, two auxiliary classifiers (small full CNNs) are added to two of the middle blocks in an attempt of combat lower layers discrimination. During training, the loss calculated from the output of this auxiliary classifier is backpropagated, while being weighted at 0.3 and during inference, it is discarded (Szegedy et al. 2015a).

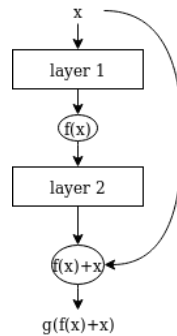
As mentioned in Section 2.2 later many different variations have surfaced with added or improved features (Jin et al. 2016, Szegedy et al. 2016a 2017a).

Resnet: Soon after InceptionNet, in 2015 a new approach to convolutional neural network architecture was suggested by researchers at Microsoft. He et al. (2015) came up with an architecture called Resnet (Residual Network). The name came from a specific part of the architecture, which is the residual block(Figure E.2).

With the hypothesis that the residual mapping will be easier to optimise than the original one, the authors achieve state-of-the-art results on multiple datasets and benchmarks. When scaled, the Resnet architecture includes multiple residual blocks, which are convolutional layers, but with "shortcut connections" as described in the original paper (He et al. 2015). This approach helps with the problem of *degradation*, which denotes the phenomenon that sometimes deeper networks perform significantly worse than narrower networks for the same problem even after converging (He et al. 2015). Resnet is a type of architecture and encompasses multiple variations of Resnet exist based on the number of layers in the architecture. For example, Resnet-18, Resnet-34 and Resnet-101 are such variations that have 18,34 and 101 layers accordingly, but they all have residual blocks. The residual blocks also vary a little with the number of layers because for architectures of 50 layers and more, a bottleneck building block is used, which differs from the original residual block by using **3** layers instead of 2.

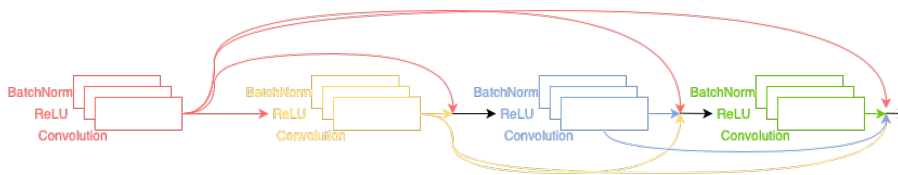
DenseNet: Fast-forward to 2018, the team of Huang et al. (2017) built on the idea of skip connections (introduced in Subsection E). They discovered a CNN built with all layers interconnected with each other. In contrast to ResNets, DenseNets do not apply summation. Instead they concatenate the feature maps produced by the different layers (Huang et al. 2017). So rather than just implementing skip connections, DenseNets implement "dense connections" as represented in Figure E.3. This way Huang et al. (2017) argue that with fewer parameters CNNs can learn faster since the need for redundant feature maps is minimised and also all information is preserved throughout the whole network,

Figure E.2: Residual block used in He et al. (2015). The identity of the input is added together with the output of *layer 2* (in this example before the next activation is done.)



which alleviates the vanishing gradient problem otherwise encountered with CNNs. Also, recent findings show that there is a significant amount of layers, which do not contribute much to the final prediction of the neural network and can be randomly dropped (Huang et al. 2016). Another key characteristic of DenseNets is that they are relatively narrower than other CNNs since they use around 12 filters per layer and the authors argue that in this way each layer contributes a small portion of the “collective knowledge” (Huang et al. 2017). Some of the other listed benefits in the paper (Huang et al. 2017) are that the dense connections have a regularisation effect, helps with improved flow of information and gradients, and contribute to better parameter efficiency. Between the dense blocks (like the one in E.3) the network has what they call a “transitional layer”, which consists of a 1×1 convolution and a pooling with stride 2 (in the case of ImageNet DenseNet the pool is average 2×2 pooling).

Figure E.3: Dense block with 4 layers used in Huang et al. (2017). Each layer outputs its feature map to each following layer in the neural network which has a matching feature-map size.



U-Net: Ronneberger et al. (2015b) takes an unconventional, at the time, approach and uses two mirrored pathways for first “capturing context” and then expansion to facilitate precise localization. It is based on how autoencoders (discussed in Section 2.1) work. The presented training strategy relies heavily on data augmentation as mentioned in the paper. In difference to all earlier presented state-of-the-art models, U-Net’s purpose is not to achieve image classification, but rather image segmentation, thus it has a relatively higher computational complexity, but is used to solve a different problem. The authors also believe that this approach “can be applied easily to many more tasks” (Ronneberger et al. 2015b).

Backbones: As computer vision is such an enormous field and encompasses so many different problems, approaches and techniques it is particularly difficult to generalise all concepts as applicable to each problem, but rather they should be studied separately. Yet, there is an immense transferable knowledge that can be applied from one computer vision problem to another (Torrey and Shavlik 2010) . One concept that exploits this postulate is the use of backbones. Backbones are neural network architectures or parts of architectures, which are taken and used as part of some larger pipeline (Ren et al. 2015) . An example of such an application is the use of ResNet (He et al. 2015) backbones in the construction of UNets (Huang et al. 2020) or the use of InceptionNet (Szegedy et al. 2015b) and ResNet (He et al. 2015) as part of object detection, where they are only used to classify the object in a region, which is separately discovered by a regression model (in the case of R-CNN with such backbones) (Li et al. 2018) . In essence, the use of such backbones is not only important to make a certain approach work. Regardless of whether it is an autoencoder task, semantic segmentation, object detection or even generative adversarial network, it also signifies that computer vision advancements and algorithms can be used to address different problems by applying predefined adjustments. Thus, heuristic approaches can exploit this and produce novel general-purpose computer vision models.

R-CNN: The field of deep learning for computer vision has been always influenced by visual neuroscience theories, yet the perspective on image processing in these fields is still substantially different (Riesenhuber and Poggio 2000) . That is why object detection before 2014 was mostly performed using HOG(Dalal and Triggs 2005) and SIFT(Lowe 2004) features, which were inspired by complex cells in V1¹(Girshick et al. 2014). However, it is also known that object recognition happens in a later stage, which according to He et al. (2017) suggests that there might be multi-stage, hierarchical processes. Based on this idea, the authors build what they refer to as "the bridge between image classification and object detection", which is the R-CNN approach. The approach uses region proposals to recognise a region where an object is present and then CNN is used to classify the object.

R-CNN is composed of three sequential methods to get from the raw image input to its final output. The first one is region proposal one, which generates 2000 category-independent regions. Then, a large CNN (usually a whole state-of-the-art CNN architecture) is responsible to extract fixed-length features from each proposed region (Girshick et al. 2014). The last stage uses class-specific linear SVM to confirm the object's presence in the proposed region (Girshick et al. 2014).

With this approach, the authors have achieved a 30% improvement from previous best results on the VOC 2012 challenge with their mean average precision score of 53.3% (Girshick et al. 2014). Later, multiple different variations emerged with Fast R-CNN (Girshick

¹V1 refers to "the first cortical area in the primate visual cortex"

2015), Faster R-CNN (Ren et al. 2015), Mark R-CNN (He et al. 2017) and others.

YOLO: After R-CNN, a new approach appeared, which was named: "You Only Look Once" or YOLO for short (Redmon et al. 2016). This approach is specifically unique because it encompasses everything in a single end-to-end neural network, which achieves state-of-the-art competitive results while being extremely fast (Redmon et al. 2016). Instead of using a sliding window or the R-CNN region proposal approach, YOLO uses features from the whole image to predict each bounding box Redmon et al. (2016). First, the input image is resized to 448x448 pixels. Then, the CNN is ran and decision to keep or discard the prediction is taken using a specified threshold. YOLO was particularly better than other approaches for false positives from image background since it uses the entire image when generating predictions instead of looking at just a region of the image. Even though the original YOLO was less accurate than the state-of-the-art, it was superior to other approaches for its generalisation capabilities for object representations (Redmon et al. 2016).

The structure of YOLO is inspired from GoogLeNet(Szegedy et al. 2015a) (Discussed in Section E) and it uses 3x3 convolutions with 1x1 reductions. YOLO has 24 convolutional layers and then two fully connected layers with final output of 7x7x30 tensor predictions (Redmon et al. 2016).

As with other models, YOLO also has multiple versions: Fast YOLO, YOLOv2, YOLOv3, YOLO9000 and others.

SSD The same year as YOLO was published, Liu et al. (2016) introduced the SSD: Single Shot Multibox Detector, which achieved better accuracy than YOLO and even a comparable Faster R-CNN model for the VOC2007 test problem. Similarly to YOLO, SSD is also a single deep neural network that achieves object detection. However, the way SSD generate the boxes is different. SSD uses the idea of default boxes over different scales and aspect ratios of feature maps (Liu et al. 2016). The algorithm first creates default boxes, which are various permutations of different sized boxes at given locations in the feature maps. Then, using "matching strategy" (Liu et al. 2016), the ground truth boxes are matched to a corresponding default box using Jaccard overlap. The authors have chosen VGG16 (explained in section 2.2) as a base network and then removed the image classification layers, then added more convolutional layers to improve predictions for different scales and finally added auxiliary structure to facilitate object detection (Liu et al. 2016). SSD feature maps from several different layers to handle objects of different scales and uses 8x8 and 4x4 feature maps.

FPN: Feature pyramid networks(FPNs) were introduced by the Facebook AI research team (Lin et al. 2017a). Although the feature pyramids were used with some of the first object detectors (Dalal and Triggs 2005, Lowe 2004), they were soon deemed computationally infeasible for use in deep convolutional neural networks. With FPNs Facebook researchers came up with a way to use feature pyramids (similar to the SSD approach

described above in section E) with marginal extra cost. Thus, they created a model, which achieved state-of-the-art performance for single-stage models on the COCO 2016 challenge. Pyramids are scale-invariant, because of the multiple levels in the pyramid, which allows for the detection of objects of various scales. The goal of this approach was to "leverage the pyramidal shape of a ConvNet's feature hierarchy while creating a feature pyramid that has strong semantics at all scales" (Lin et al. 2017a). The network uses two pathways- bottom-up and top-down, which are responsible for computing feature maps of the backbone CNN and estimation of higher resolution features respectively.

The bottom-up pathway has a scaling step of 2. It is used only for the last outputted feature map of each *network stage*². The top-down pathway uses nearest-neighbour upsampling to increase the spatial resolution of the feature map by a factor of 2. A 1x1 convolution is applied to the output of the bottom-up pathway to reduce the channel dimensions and then it is combined with the top-down pathway to achieve the goal of the paper mentioned above (Lin et al. 2017a).

RetinaNET: In 2017 the research team of Facebook Lin et al. (2017c) published a paper, introducing the RetinaNet. The authors argue that the single-stage approach may potentially be faster and simpler than the R-CNN two-stage approach. One of the biggest contributions of the paper is the revised cross-entropy loss function. The proposed novel loss is called focal loss, which aims to combat the extreme foreground-background class imbalance found in training sets for object detection (Lin et al. 2017c) but it is extendable to classification problems as well.

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \log(p_t) \quad (\text{E.1})$$

In the equation above p_t refers to the model's probability for the class with corresponding label y such that:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

. The parameters α and γ represent the weighting of different class and the focusing parameter respectively. The focusing parameter in particular smoothly adjusts the at which the "easy" samples are downweighted. The authors argue that the conventional cross-entropy loss has a pitfall when used in object detectors because the easily classified negatives dominate the gradient and make up the majority of the loss, which is why they have created a new focal loss (Eq. E.1) (Lin et al. 2017c). The γ parameter is responsible for modulation, thus the bigger this focusing parameter - the more down-weighted the easy examples are. The α parameter is a weighting parameter introduced to achieve an α -balanced cross-entropy loss (Lin et al. 2017c). The detector (RetinaNet)

²A network stage is defined by the authors Lin et al. (2017a) as any group of layers within a network, which output a feature map of the same size

uses the focal loss and adopts a feature pyramid network backbone Lin et al. (2017a) with 2 subnetworks responsible for classification and box regression respectively. For the classification, there is a fully-connected network attached to each level of the FPN. The subnetwork is composed of four 3x3 ReLU activated convolutional layers with filters, which number correspond to the number of channels of the input feature map. Then a 3x3 convolutional layer with filters equal to the number of classes multiplied by the number of anchors. The last layer is a sigmoid or softmax layer depending on if the problem is binary or multi-class (Lin et al. 2017c).

The second subnetwork is used for box regression and is also a fully-connected network, which is also attached to each level of the FPN (Lin et al. 2017c). It regresses the offset between each anchor box and a nearby ground-truth object. The structure of the network is the same as the object classification subnetwork with a single difference. The last layer has a linear output and the number of neurons in it is 4 times larger (because of the 4 points needed to construct the box) than the number of anchor boxes (Lin et al. 2017c).

EfficientDET: While RetinaNET achieved new state-of-the-art performance thanks to the optimisation of floating-point operations and managed to stay hugely competitive and outperform many of the previous approaches, EfficientDet takes object detection efficiency one step further by exploring two main ideas: weighted bi-directional feature pyramid and compound scaling method (Tan et al. 2020). The motivation behind Efficient-Det was that the new NAS models such as AmoebaNET (discussed in Section F) were becoming bigger and bigger and "the large model sizes and expensive computation costs deter their deployment in many real-world applications" (Tan et al. 2020). This is a huge problem because even though these models are highly accurate they have a hard limit on the applications they can be used in. What EfficientDet does differently than normal FPN approaches (following the methodology discussed in E), is that the authors weigh the features coming from different resolutions of the pyramid, hence achieving weighted bi-directional feature importance (Tan et al. 2020). In this way, they introduce new parameters which are responsible for capturing the importance of different features. The second advantage which allows EfficientDet to surpass state-of-the-art performance while having a magnitude fewer parameters lies in what they call 'compound scaling' (Tan et al. 2020). It scales up the resolution not only of the backbone but also the feature network and the box or class predictor.

Hide-and-Seek: With the increased interest in the field of object detection a need to upscale object detection datasets started to arise. The need for an upscaling augmentation which could boost generalisation performance as well as size of the datasets was needed (Singh and Lee 2017b). In (Singh and Lee 2017b) authors addressed just that by coming up with the idea to hide salient parts of the image during training, thus forcing the model to look elsewhere to find features. Thus, the network can learn much more about

objects and their properties. There is also a reported significant increase in performance for object localisation compared to the state-of-the-art in 2017. This is achieved with a weakly-supervised method, which aims to localise all salient parts of an object, instead of just looking for the most discriminative part. During training, it hides patches of the image to hide the most discriminative parts and during inference, no augmentation to the images is done. The method can mainly be applied to two general scenarios: object localisation in images and action localisation in videos (Singh and Lee 2017b), succeeding in improving the state-of-the-art object localisation score on ILSVRC. During the development of the method, the authors encountered two interesting problems. The first is that a discrepancy between the neurons in the first convolutional layer starts to appear since the filters in the first convolutional layer are exposed to lots of empty pixels, thus distorting the underlying distribution. This affects negatively the performance when the method is tested for inference since during inference the images are intact (Singh and Lee 2017b). To combat this, the authors use to fill the removed patches with the mean vector for RGB values over the whole dataset. The second interesting issue they tackle is the normal performance of action localisation methods. During action localisation algorithms usually focus on the most discriminative frames to ensure good performance and usually that leads to many frames being missed out. Singh and Lee (2017b) removes random frames during training (sampled from different segments with a predefined probability). This way the algorithm can be exposed to different frames and thus learn to localise action better following the same logic as the application for object localisation.

Appendix F

State-of-the-Art Neural Architecture Search Methods for Computer Vision

AmoebaNet Noticing the trend in computer vision and the effort towards a general purpose AI, back in 2017 the Google Brain Team developed an algorithm based on neuroevolution (Real et al. 2017)- AmoebaNet, in line with the trends in computer vision . The overall aim was to achieve automatic neural architecture search through employing neuroevolution following a simplistic evolutionary approach backbone. The training was via backpropagation with a standard optimiser, but an intriguing novelty was the use of a simplified graph to represent architectures during encoding, enabling the approach to use mutation on each node separately with the ability to occur multiple times (Real et al. 2017). In their study, the authors compared the approach to random search as well as reinforcement learning methods. The evolutionary approach seemed to outperform all other approaches. Although their performance has been currently surpassed by EfficientNet (Tan and Le 2019b), the AmoebaNet is still considered to be state-of-the-art. In the last couple of years, there have been several further improvements to the initial model, thus several versions of the same model currently exist. All of them are grouped in a family, which is called AmoebaNets, so each improvement has a specific suffix (ex. AmoebaNet-A, AmoebaNet-D etc) (Real et al. 2017) (Real et al. 2018).

NASNET: While researchers at Facebook (He et al. 2017) and Google (Real et al. 2018, Tan and Le 2019b) were focusing on solving image classification, semantic segmentation or even instance segmentation problems, Qin and Wang (2019) decided to look at a very different computer vision problem. They examined the detrimental effect on images when there is rain (or similar artefacts) and decided to come up with a novel attention mechanism to combat it (Qin and Wang 2019). The idea behind their approach was that it re-calibrates neuron-wise feature responses by observing codependence and

correlation between neurons. In contrast to early methods that focus on image priors (Sun et al. 2014, Kang et al. 2011), they fuse information of peripheral neurons allowing them to capture the attention of local receptive fields as well as contextual information. One of their key contributions is the Neuron Attention module of their approach. It is meant to be adaptive and dynamic in re-calibrating responses of neurons by modelling their inter-dependencies and influence over each other (Qin and Wang 2019). They achieve this with two operations: depth-wise convolutions because of the extra spatial information per channel and point-wise convolution to tackle the problem with the effectiveness of depth-wise convolutions to capture the information of the different filters in the same spatial location (Qin and Wang 2019).

DARTS: Arguably one of the most important developments in Neural architecture search is DARTS (Differential Architecture Search) (Liu et al. 2018). In contrast to conventional evolutionary or reinforcement learning approaches, the researchers involved in this paper use continuous relaxation of the search space for architectures, which allows for efficient search using gradient descent. The work discusses one of the fundamental challenges with neural architecture search approaches - scalability (Liu et al. 2018). While most NAS approaches treat the optimisation of neural networks as a black-box optimisation, in contrast to DARTS which applies gradient descent optimisation over a continuous search space of architectures and while the continuous search space relaxation is not novel by itself, DARTS aims to find complex building blocks with convoluted graph topologies which are not restricted to any family of blocks (Liu et al. 2018). In their work, the authors search for a computational cell, which is represented by a directed acyclic graph consisting of an ordered sequence of nodes. To relax the choice of layer operations to be continuous the authors use a softmax function applied over all possible operations while at the same time attempting to optimise the weights of the network, which results in a bi-optimisation problem (Liu et al. 2018). In their experiments, they use a two-stage process where they first find repeatable cells (with 7 nodes) and then they stack some of these discovered cells to create an architecture which they compare with the state-of-the-art.

DEvol: Deep Neural Network Evolution (DEvol in short) is a generic framework for genetic optimisation of neural networks (Davison 2017). In reality, even at first sight simple, the approach manages to construct a full-fledged neuroevolution process and achieves above 99.3% on the MNIST dataset without using any transfer learning, data augmentation, ensembling or even fine-tuning. The authors run their approach for 50 generations with a population size of 20, which results in 1000 model construction and evaluations (Davison 2017). DEVol also uses early stopping, parameter selection and a limited number of epochs (in their experiments 10) to avoid using computational resources for genomes that are not promising. They employ one-point crossover and for their mutation, they change the value of a random parameter with the one from a pool of possible

values (Davison 2017).

EfficientNet: While the team of Google Real et al. (2017) managed to set a new state-of-the-art performance model and even further improve it just a year later (Real et al. 2018), researchers from the same research group decided to pursue a similar ambition but instead of focusing on neuroevolution, they employed reinforcement learning (Tan and Le 2019b). They base their approach on the idea that manual scaling of two or more dimensions is possible, yet requires a tedious amount of labour and yields sub-optimal results (Tan and Le 2019b). Hence, the authors attempt to automate this process using reinforcement learning and rethink the way CNNs are scaled up. At first, they apply their new scale-up method to already existing architectures but then decide to employ neural architecture search to discover a new building block for CNNs which at that time set a new state-of-the-art performance (Tan and Le 2019b).

MONAS: Sometime before EfficientNet and slightly after the improvement of Real et al. (2017) procedure in the face of Real et al. (2018) came to light, the success of automated neural architecture search was spotted by the authors of Hsu et al. (2018). They managed to create an algorithm called MONAS (Multi-objective neural architecture search) which was one of the first attempts in incorporating multiple (often conflicting) objectives in neural architecture search (Hsu et al. 2018). Their study focuses on the idea that a correlation between model complexity and achieved accuracy has been observed and while highly accurate models are generally preferred, there are settings in which the model complexity makes it unfeasible for deployment as a solution (Hsu et al. 2018). This is why they introduce an objective based on the number of operations and predicted energy consumption while searching for an optimal architecture (Hsu et al. 2018). The authors utilise reinforcement learning with long short-term memory cells which control the hyperparameters for different layers (Hsu et al. 2018), but their approach differs from similar related work thanks to the fact that they observe that while using multiple objectives, their algorithm manages to produce solutions which satisfy specified constraints much more effectively than random search (Hsu et al. 2018).

AutoDeepLab: While the methods discussed so far focussed on simple classification, in 2019 ambitious researchers from Google and John Hopkins University Liu et al. (2019a) decided to explore the possibility of using the already popular Neural Architecture Search (NAS) methods to discover an architecture for semantic segmentation. They establish that despite most of the knowledge from conducting NAS on other problems being relevant, some assumptions do not necessarily work in the case of semantic segmentation. They recognise there are two big problems with NAS and these are: 1. the need for a more relaxed and general search space and 2. better efficiency for performance estimation since state-of-the-art NAS approaches can take extremely long times to discover a competitive architecture (Liu et al. 2019a). Instead of focusing on Neuroevolution (like AmoebaNet) or reinforcement learning (like EfficientNet), Liu et al. (2019a) employ

gradient descent to optimise both weights and architecture. One of the key assumptions they centre their work on when looking for architectures is that the current state of the art utilises a two-level hierarchy (He et al. 2016b, Szegedy et al. 2015b, Liu and Deng 2015). While most NAS approaches do focus their efforts to match these manually designed architectures (by incorporating macro and micro search) (Lu et al. 2019), Auto Deep-Lab propose a trellis-like network-level search space to form hierarchical architecture search Liu et al. (2019a). They follow differentiable NAS formulation (Shin et al. 2018) and use stochastic gradient descent to beat the Cityscapes benchmark and achieve 80.33% mean IOU on the dataset using 20 out of the 30 classes. The paper also argues in favour of optimising the whole architecture instead of conducting micro and macro search separately, which is rare in the field of NAS.

Appendix G

Datasets

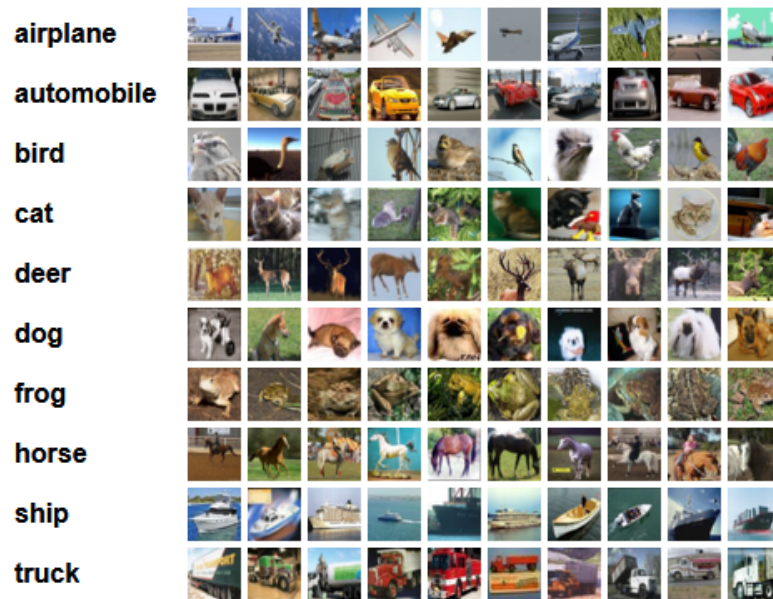
In this appendix, some popular image classification problems/datasets are presented, used in the following chapters. In addition, a new, original dataset specifically collected for this thesis is shown, which will be introduced in chapter 5. Image classification is one of the most mature, simple and fundamental computer vision problems (Nath et al. 2014). It is usually associated with a problem where a single image should be classified to belong to one of two (binary classification (Parkhi et al. 2012)), one of many (multiclass classification (LeCun et al. 1999)) or multiple different (multilabel classification (Miao et al. 2019)). Here some of the quintessential image classification datasets are presented.

CIFAR10 (Krizhevsky et al. 2014) is a dataset composed of 10 classes, which are aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Usually, the dataset is used as-is and the task for the dataset is image classification where each image is classified as being a single class out of the ten. There do exist extensions of this task, however, which group different classes together and redefine the task (Wan et al. 2020, Ma et al. 2021).

For example, instead of predicting if the image is either of the ten classes, the problem can be structured, so that a model needs to predict whether an image depicts a vehicle or an animal (Hussain et al. 2018). In this way, the actual classes are grouped, but the original labels can be used as concepts (Kazhdan et al. 2020, Sarkar et al. 2022) and in this way, this extra information can be supplied to a model to explore representation learning and other interesting ideas. The same thing can also be done the other way around, instead of the concepts being the actual classes and the labels being the groupings both can be reversed (Ma et al. 2021).

Doing so, allows the model to have information if the image represents an animal or vehicle but it still has to guess which one it is. This also opens the door to various active learning research opportunities and exploration as to what part of the data is most important for learning. This dataset is widely used in various computer vision problems, benchmarks and scenarios and is believed to be a harder toy dataset than MNIST (LeCun

Figure G.1: Example images with all 10 classes of CIFAR10 (taken from <https://www.cs.toronto.edu/~kriz/cifar.html>) (Krizhevsky et al. 2014)



et al. 1999), which allows for more accurate performance estimation for real data.

ImageNET: Arguably one of the biggest computer vision datasets and a driving force for the rebirth of the field of computer vision was the creation of ImageNet back in 2009 (Deng et al. 2009). ImageNet is a dataset which started with around 20 000 images of different everyday objects and with the help of an annually held competition called ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al. 2015) was expanded over the course of 7 years to become one of the largest annotated image datasets in the world. It contains 14 million images classified into more than 21 thousand classes and subclasses. The classes are structured using the popular framework WordNet which is a large lexical dataset (Fellbaum 2010). Many researchers use ImageNet to benchmark general models, approaches and novel ideas and it is also of interest to researchers in the area of active learning (Emam et al. 2021) and AutoML (He et al. 2021).

SIXray: In the paper Miao et al. (2019), the authors present a new publicly available security threat dataset made up of more than a million X-ray scans of baggage. The data is split into 7 classes (Gun, Knife, Wrench, Pliers, Scissors, Hammer and negative), but because the Hammer class is made up of merely 60 images they didn't use it in their experiments. The dataset itself is the biggest dataset for baggage scans to date and the data was collected from several subway stations (Miao et al. 2019). In the paper, the dataset is described to have the following 4 important properties: 1. Presence of overlapping objects 2. Inherent intra-class variation from the different shapes, scales, viewpoints and styles 3. Noise from the heavily cluttered objects and 4. Heavy class

imbalance.

SIXray consists of 3 separate datasets: SIXray 10, SIXray 100 and SIXray 1000 denoting 1:10, 1:100 and 1:1000 ratio of threats to benign images respectively. This means that in SIXray 1000 for every image of a prohibited item there are 1000 benign ones, which can easily overwhelm a classifier and the class imbalance has to be considered when designing a proper approach. More specifically, when choosing the right loss and metrics.

For this reason Miao et al. (2019) introduces an approach, which they call class-balanced hierarchical refinement (CHR). This method intends to combine two sources of information and calculate a loss. It takes into account the different level features by examining multiple different layers, which are chosen as feature extractors. The feature maps produced from each chosen layer are then upsampled to the original size of the image (if it is of a different size) and concatenated with the feature maps from the previous layers. Then the result is fed into a refinement function before being sent for classification.

Object Localisation: In difference to pure image classification, where one image is just associated with a label, the object localisation task takes this one step further by requiring the system to provide information not only by a single label, but rather give context as to where the actual entity is positioned (Tompson et al. 2015) . In contrast to object detection, however, object localisation is just concerned with finding one particular entity, thus is more similar to image classification with some auxiliary regression over 4 parameters which define a single bounding box (Harzallah et al. 2009) .

Appendix H

Anomaly detection

Anomaly detection deals with the "detection of deviation and divergence of anomalous samples from the normal ones" Minhas and Zelek (2019). It is concerned with recognising patterns of data (called anomalies), which appear to be out of the ordinary distribution (Chandola et al. 2009). Recognising these anomalies is crucial for various problems from a multitude of domains, which range from fraud detection, surface defect detection, diagnostics and even threat detection (Minhas and Zelek 2019).

Identifying anomalies can be especially hard in scenarios where the natural distribution of data involves high variance (as in credit card fraud detection) since most of the anomalies can be well concealed to look like natural outliers of the original distribution (Mery 2015). Taking anomaly detection one step further is of extreme importance in multiple different industries such as medical imaging (Wei et al. 2018), manufacturing (Mery 2015) and concealed weapon detection (Miao et al. 2019).

There exist anomaly detection algorithms that make use of supervised, semi-supervised and unsupervised learning, where unsupervised learning techniques are believed to still be unavailable for analysing images (Minhas and Zelek 2019).

Using supervised learning for anomaly detection there are two major challenges: the "lack of labelled data and low anomaly instances" (Minhas and Zelek 2019)

The lack of labelled data can be combated either on the data level (by manipulating or adjusting the data) or as part of adjustments to the mathematical model used to discover the anomalies.

In the "data space", a popular way to deal with huge data imbalance if possible is to over-sampling the data in favour of the misrepresented class or classes using approaches such as SMOTE (Chawla et al. 2002), MWMOTE (Barua et al. 2012) or many others. As a result, new data points are generated which follow the distribution of the presented dataset. With imagery data oversampling is usually done using data augmentation (Taqi et al. 2018). Data augmentation is a great tool which allows one to specify hyperparameters which in the case of images might be rotating the image at certain degrees, flipping the image along each axis, brightening or darkening the image etc.

In the "model space", there also exist numerous different techniques to address limited data availability and data imbalance (Miao et al. 2019). If the data follows similar distribution even partially to an already existing method or dataset then transfer learning can be applied Minhas and Zelek (2019). Using this method the model can be fine-tuned to the dataset at hand while using an already trained model for a different dataset (Ver-cruyssen et al. 2017). A dataset often used for transfer learning is ImageNet (Section G) since it is one of the largest image datasets available (Deng et al. 2009). To apply transfer learning researchers and practitioners use a model which is already good at solving a particular problem (in this example ImageNet) and then retrain it on the newly presented problem(dataset) (Torrey and Shavlik 2010). This allows for the model to start at a potentially good starting point while training on the new data points to reach global optimum more efficiently and effectively.

Another technique used to combat class imbalance is adjusting the biases of the output neurons for some of the classes (Givnan et al. 2022) or using weights in the loss functions to steer the model in giving underrepresented classes more attention (Zhu et al. 2018) as well as using loss functions like Focal loss (Lin et al. 2017b) which are specifically designed to be as invariant as possible to class imbalance. These techniques which in this work is called 'loss manipulation' presents a way to specify preference articulation and make the model give a certain class more attention to other ones and in this way the training of the model can be steered into gaining bigger rewards in minimising the loss when the performance for certain class or classes is improved. Below an example of how the weight coefficients are added in normal categorical cross-entropy is presented. Unweighted crossentropy can be represented by the following formula:

$$-\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C 1_{y_i \in \mathbf{C}_c} \log p_{model}[y_i \in \mathbf{C}_c]. \quad (\text{H.1})$$

There is a double sum to calculate the loss over each observation i from all observations N and each class c from all C classes. The term $1_{y_i \in \mathbf{C}_c}$ is used to describe the indicator function for selecting the i^{th} element that belongs to class c . The probabilities for each observation i to belong to class c as predicted by the *model* are $p_{model}[y_i \in \mathbf{C}_c]$, where \mathbf{C} is a C -dimensional vector of probabilities for each class c . Such probabilities are typically weighted (e.g., to assign a higher importance to the true positives of a class with respect to the rest), and expressed as \hat{y} , and thus the loss L becomes:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C \alpha_c 1_{y_i \in \mathbf{C}_c} \log \hat{y}, \quad (\text{H.2})$$

where y contains all the labels, \hat{y} the model predictions for a particular instance y_i belonging to any of the classes c , and α_c is the weight of the particular class.

Semi-supervised approaches attempt to estimate the underlying distribution and den-

sity function (Akçay et al. 2018). While there exists a plethora of semi-supervised approaches one particularly interesting and successful is using generative adversarial networks to tackle this problem (Akçay et al. 2018). This approach attempts to first minimise the difference between the images from the dataset and their generated reconstructed counterparts. This way a larger distance from this learned data distribution at inference time "is indicative of an outlier from that distribution" (Akçay et al. 2018), that is, an anomaly.

Appendix I

Encoding convolutional neural network architectures

A prerequisite for using evolutionary algorithms to solve a particular problem is to provide the algorithm with a search space of decision variables, which mapped through an evaluation function (or functions) produces a result in the objective space. To do this with neural architectures multiple different approaches can be undertaken (like using cellular encoding (Gruau 1994b), genetic encoding (Dürr et al. 2006) and many more). Usually, the encoding techniques are split in direct encoding (which means directly mapping the genotype to a single phenotype) (Ronald and Schoenauer 1994) or indirect encoding (which involves mapping the architecture and other hyperparameters through some function, which produces a summaries version of the original information and serves as a DNA of the actual variables (Stanley 2003)). In this experiment, a direct encoding approach is discussed, but expanding that to an indirect one with the motivation to reduce the search space could also be explored.

One of the major stepping stones in the process of constructing a neuroevolution algorithm is the encoding and decoding of the neural network. Thus, this subsection covers the progress in developing the neuroevolution approach discussed in the aims and objectives and future work. Drawing motivation from the encoding scheme of the paper "Darts: Differentiable architecture search" (Liu et al. 2018) (DARTS for short), my current novel approach is capable of encoding convolutional neural networks using 3 python lists for convolutional and dense layers as well as a single value (identifier) of the desired global pooling approach. For successful construction of the CNN, the shape of the input, as well as the number of classes, is also required. The 3 required lists are as follows:

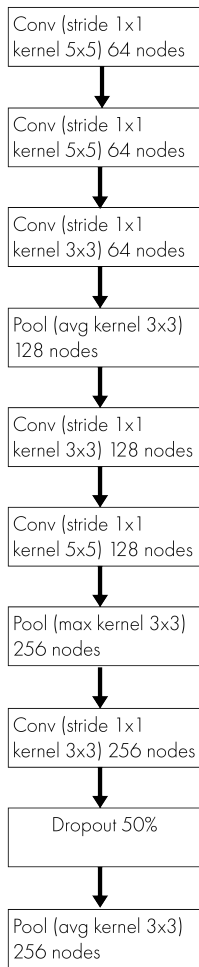
1. A list of all layer ids, where a dictionary of id to layers is predefined (ex. 1- > 3x3 convolution)
2. A list of number of nodes per each layer

3. A list with integer values corresponding to binary values for each layer. This input is then taken and converted to its binary representation, which is used to construct a directed acyclic graph.

The way the encoding is structured allows for the use of cells (similar to the approach of DARTS (Liu et al. 2018)). These cells are abstract separations of a set of layers. For convenience, the convolutional and pooling layers are separated from the dense layers after the flattening or global pooling in different cells and each cell consists of a 3-D tensor containing the 3 lists from above. As evident from Figure I.1, any set of forward connections in a cell can be encoded, while a drawback is that no skip-connections can be made between cells. This allows for great modularity and in the context of an evolutionary algorithm, this allows for cells to be easily swapped to facilitate recombination of different genomes.

This experiment is a major stepping stone in the creation of a neuroevolution algorithm and is subject to change. However, it is proof of progress towards objective 3 and enables great flexibility for the further developed neuroevolution algorithm. The next major efforts will be into creating an initialisation, which is also supported by literature and adding the weights and biases to the encoding. A fast experiment for using the random seed for normal initialisation was attempted (as a mean of indirect encoding), but it was abandoned because of the lack of literature to support the feasibility of the idea as well as the implicit lack of robustness.

Figure I.1: Illustration of the encoding/decoding process. a) presents the same architecture described in b), but a) is decoded and b) is encoded. c) presents a more in-depth view of how the connections array from b) is decoded and d) is an example b) of a connections array which is [128,112,33,16,8,4,2,1]

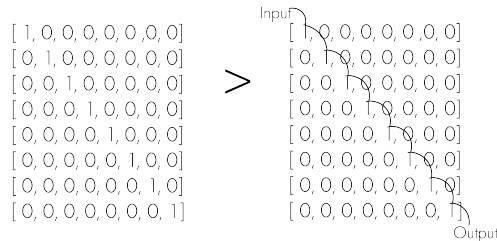


a) Decoded cells

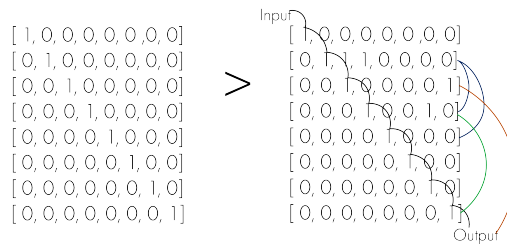
b) Encoded arrays

Layers	[3 , 2 , 30 , 2 , 5 , 35 , 2 , 550 , 30]
Nodes	[64 , 64 , 128 , 128 , 128 , 256 , 256 , 0.5 , 256]
Connections	[128 , 64 , 32 , 16 , 8 , 4 , 2 , 1]

c) Triangular connection matrix representing DAG



d) Example of triangular connection matrix representing DAG with more connections



Appendix J

Concealed Weapon Detection using State-of-the-art Object detection

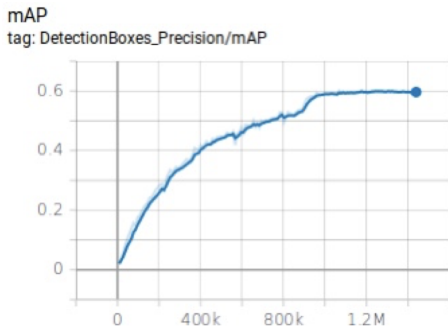
The motivation behind this experiment is to establish the performance of state-of-the-art models for object detection on concealed weapon x-ray dataset.

The authors of (Miao et al. 2019) provide a limited number of bounding box labelled images and not all threat images have an assigned bounding box for the threat, thus to evaluate the effectiveness of this approach the CNNs were only tested on images with bounding boxes. This means that the algorithms were not tested on fully negative images and to accurately measure the false positive rate this has to be done.

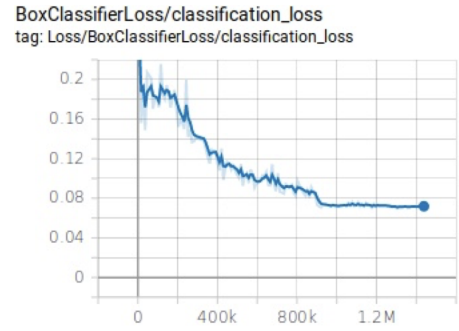
That is why one of the first conducted experiments was the use of state-of-the-art object detection algorithms on SIXray(Miao et al. 2019). In object detection, an often-used performance measure is the intersection-over-union (IOU) (Rezatofighi et al. 2019). During this experiment, Faster R-CNN (Ren et al. 2015) was used with InceptionResnet backbone and the produced results after 1 million steps converged to 62.9% mean average precision across all IOUs (graph of training MAP in Figure J.1a and the loss in Figure J.1b) during training. During inference, this approach scored 60% on the validation set and 70.78% on the test set (Breakdown of the performance per class available in Figure J.1c).

From this experiment, it can be seen that the state-of-the-art models perform quite well on SIXray, which connotes that x-ray concealed weapon detection does share some similarities with normal visual data and knowledge and approaches to some extent can be transferable for this problem. On the other hand, 62.9% mean IOU is a possible point of improvement and it presents an opportunity for niche research in order to optimise these or other networks to perform better in the critical context of security.

(a) Mean average precision of training Faster R-CNN with InceptionResnet. Light blue indicates real loss values and the dark blue is the produced line after 50% softening is applied.



(b) Loss for training Faster R-CNN with InceptionResnet. Light blue indicates real loss values and the dark blue is the produced line after 50% softening is applied.



(c) Performance for different classes of the approach on the test set. The first result is the current state-of-the-art performance for COCO dataset, which is widely known object detection dataset problem. The second bar (the brown one) is the trained loss

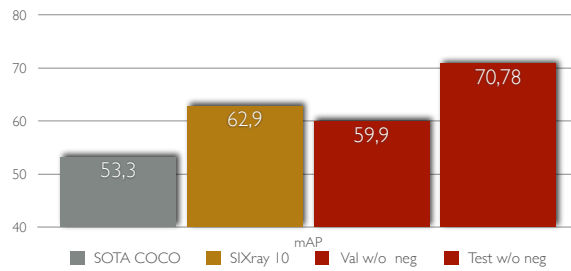


Figure J.1: Object detection experiment results. In the first two figures (a) and (b) the x axis denotes the training steps (how many minibatches the algorithm iterated over) and the y axis denotes the mean average precision in (a) and the loss in (b)

Appendix K

Preference articulation through loss manipulation

From looking at the results in the original study for SIXray (Miao et al. 2019), it becomes evident that the performance for some of the classes in the context of SIXray100 and SIXray1000 can drop to near-random and sometimes even worse. All presented approaches report the worst performance for the Scissors class, which also happens to be the most underrepresented class. It is worth mentioning that according to the code provided by the authors, the performance of for SIXray100 and SIXray1000 is reported after the algorithm is trained on SIXray10 and then inference is conducted on each of the three datasets to discover their performance. This observation connoted that there might be a correlation between the number of available samples for each class and their performance on all of the three different datasets. Some deviation in the performance is to be expected since intuitively the lower the number of samples the algorithm can observe while training the higher the probability of underfitting goes. This is because the algorithm is not exposed to all features that make up the particular class. Research in this area has currently made tremendous progress in achieving good results with less data through techniques like data augmentation (Singh and Lee 2017b) , one-shot learning (Chen et al. 2019) , zero-shot learning (Li et al. 2019) and many other interesting approaches (Ghadakchi et al. 2019, Teh and Taylor 2020). However, setting this aside, one of the main features of the SIXray dataset is that it is highly imbalanced and this is done on purpose for the dataset to be representative of the imbalance, which is usually present in real-world scenarios in this domain (Miao et al. 2019).

A series of experiments were conducted on the SIXray dataset (Miao et al. 2019) to test the hypothesis: Through preference articulation in the form of loss manipulation for adjusting how much attention is given to different classes, the performance for underrepresented classes can be improved. This is done by assigning weights to the output loss for each class. In particular, in the last layer the loss function used is binary cross-entropy,

but because it is applied to 6 sigmoid activated neurons separately (denoting if a threat from the 6 different classes is present).

In the following two equations the following convention is used:

1. C is the number of classes
2. s_i is the score output of the neural network.
3. t_i is the ground truth target value for this class.
4. w_i is the weight applied to each class.

The non-weighted version of cross-entropy:

$$\sum_i^C t_i \log(s_i) \quad (\text{K.1})$$

The weighted version of the cross-entropy:

$$\sum_i^C w_i t_i \log(s_i) \quad (\text{K.2})$$

During the setup of the experiments, some discrepancies with the original paper and methodology were found, thus, unfortunately, results from this study are not fully comparable with the original results produced by the authors of (Miao et al. 2019). The reason for this is that during the training the authors of the SIXray paper used the testing set for validation and their final results are, therefore "contaminated" with data leakage from the actual test set. In the experiment discussed below the training and validation was done on the training set and a predefined split is defined so that the training and validation splits are the same for all models. Because the datasets are imbalanced there is a high possibility of choosing a validation split where no records from a certain class are present if the sampling is done at random. That is why a stratified approach is used to ensure that the validation set contains the same percentage of records for each class. For example, if the validation set is 20%, then it will be constructed by taking 20% of the records for each class.

Afterwards, several runs of the same architecture were conducted and all hyperparameters were defined initially to be used in all experiments for all architectures. This is done in order to keep any hyperparameters except the loss class weights as control variables, so the difference between any experiment settings would be just the used architecture and if the loss manipulation is applied or not. Then, a model with the corresponding architecture was trained with the predefined architecture (mostly taken from the original paper) and the rest of the methodology has also been kept the same. After the model is trained on SIXray 10, it is then evaluated on the final testing set provided for each separate problem (SIXray10, SIXray100 and SIXray1000).

The first set of produced results looked promising since (as evident from Table K.1) for Resnet 50 and 101 the loss manipulation generally improved the performance of the classifier across almost all classes and at first glance yielded better performance even from the state-of-the-art described in Miao et al. (2019). Interestingly, after executing 15 independent runs the results were drastically different when using average precision (instead of a cherry-picked threshold) as evident from Table K.2. What is even more interesting, is that looking at the 15 runs not only the loss manipulation did not affect the performance positively it, in fact, lead to a drop in performance for the important classes, which is an unexpected finding.

Even though the null hypothesis for this experiment cannot be rejected based on the results, the following key takeaways can be drawn:

1. A reason that might have caused this huge discrepancy is that the hyperparameters used (in terms of learning rate etc.) are based on the average of all optimal discovered ones, which is suboptimal and automated approaches such as neuroevolution may be able to tackle this problem.
2. The performance of the used algorithms vary greatly depending on the used random seed for the initialisation of the weights, which in theory should be one of the few points of difference between the runs. This might signify the importance of a strong initialisation and also connote that the performance of an algorithm is predetermined by its initialisation.
3. A reason for seeing such poor performance might also be due to underfitting since the deeper architectures have worse performance than the shallower ones. To the best of the author's knowledge, currently, there is no measure of estimating the needed capacity of a neural network (and the possible architectures with this capacity) solely based on the data itself. As suggested by (Goodfellow et al. 2016b) such an approach would require the complexity of the dataset to be discovered first and then from that infer the needed capacity. Research in this area can be ultimately useful for neuroevolution and other neural architecture search algorithms since if the optimal capacity of a classifier is known, then the search space can be constrained and the algorithms would produce much better algorithms much faster. It can essentially serve as a preference articulation vector (Rostami 2014), thus narrowing down the search space.

Table K.1: Precision for 0.5 threshold in % for each class and macro mean precision for SIXray 10,100,1000

Method	Gun			Knife			Negative			Pliers			Scissors			Wrench			Macro avg		
Resnet34	96	86	76	90	68	63	97	100	100	73	32	15	75	21	15	49	26	23	80	55	49
Resnet34 + CW	95	76	63	85	63	56	97	100	100	73	32	13	75	16	11	47	24	21	79	52	44
Resnet50	95	74	60	86	62	54	97	100	100	75	34	15	74	22	15	51	23	19	80	52	44
Resnet50+CW	94	82	71	87	60	52	97	100	100	74	35	15	79	24	17	49	23	19	80	54	46
Resnet101	94	63	46	87	72	66	97	100	100	50	10	4	58	12	9	21	3	2	68	43	38
Resnet101+CW	97	78	63	93	57	44	96	100	100	70	30	13	77	16	10	41	22	20	79	51	42

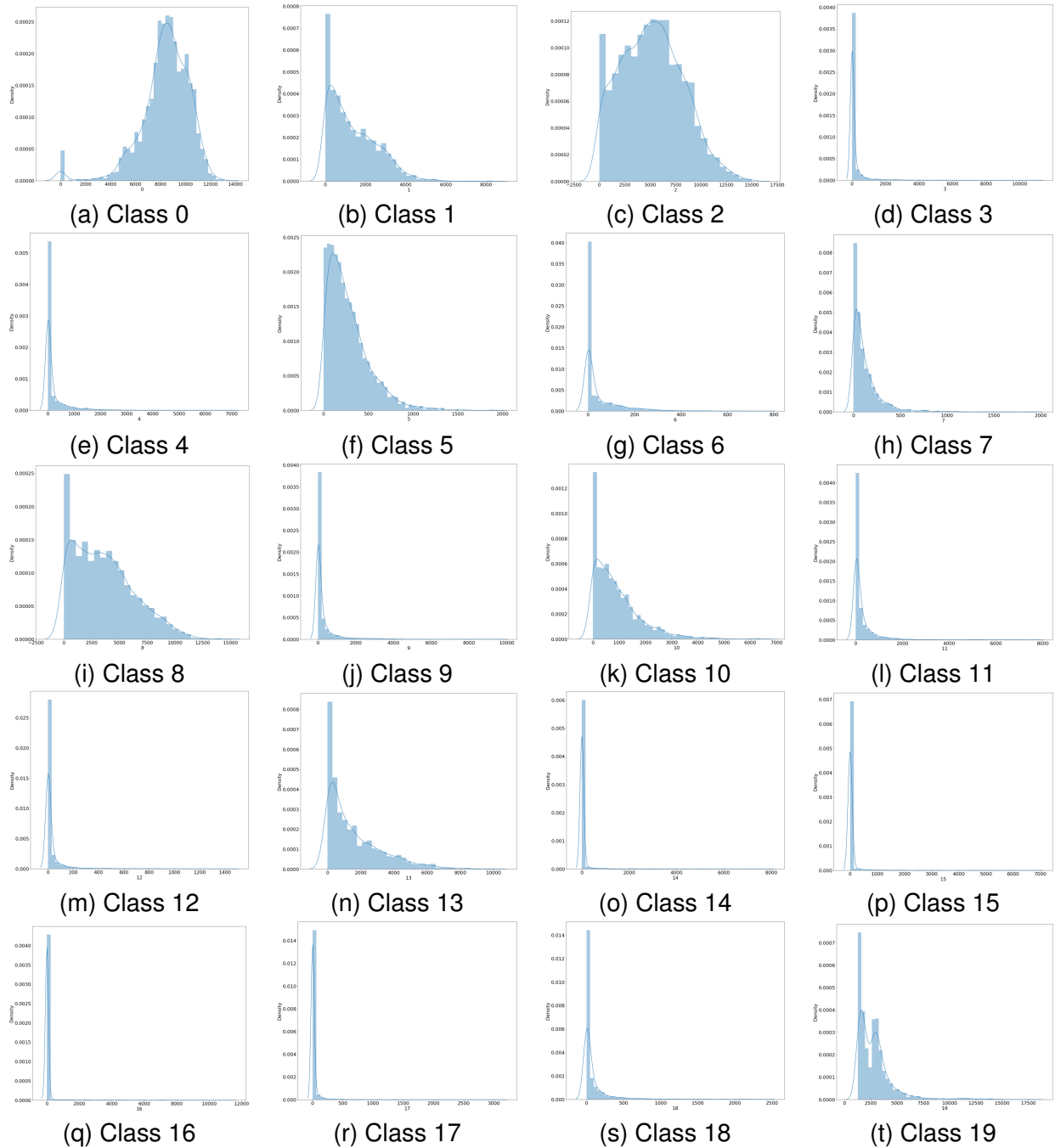
Table K.2: Average precision in % for each class and macro mean recall for SIXray 10,100,1000 for 15 runs.

Method	Gun			Knife			Negative			Pliers			Scissors			Wrench			Macro avg		
Resnet34	82 μ 2.4	65 μ 8.3	53 μ 10.7	61 μ 4.1	46 μ 5.6	43 μ 7.7	99 μ 0.1	100 μ 0.01	100 μ 0.06	51 μ 2.2	21 μ 2.6	11 μ 2.1	10 μ 0.1	1 μ 0.2	1 μ 0.1	23 μ 0.1	10 μ 1.2	8 μ 1.2	54 μ 1.4	40 μ 2.3	36 μ 2.6
Resnet34 + CW	79 μ 5.1	56 μ 12.6	44 μ 14.8	58 μ 5.9	42 μ 10.2	39 μ 11.0	99 μ 0.1	100 μ 0.05	100 μ 0.03	51 μ 1.8	19 μ 2.0	10 μ 1.5	10 μ 1.3	1 μ 0.2	1 μ 0.1	22 μ 2.2	9 μ 1.6	7 μ 1.5	53 μ 2.0	38 μ 3.9	34 μ 4.3
Resnet50	77 μ 6.6	52 μ 18.6	41 μ 20.3	62 μ 13.0	41 μ 18.0	36 μ 17.2	99 μ 4.4	100 μ 0.08	100 μ 0.05	50 μ 2.3	18 μ 4.2	9 μ 2.4	10 μ 2.3	1 μ 0.5	1 μ 0.4	21 μ 2.0	7 μ 2.3	6 μ 2.2	53 μ 3.3	37 μ 4.6	32 μ 4.7
Resnet50+CW	76 μ 8.5	47 μ 17.6	35 μ 17.1	55 μ 11.8	27 μ 15.4	22 μ 15.5	99 μ 0.7	100 μ 0.06	100 μ 0.03	50 μ 4.9	18 μ 5.1	9 μ 3.1	9 μ 1.6	1 μ 0.2	1 μ 0.1	21 μ 3.1	7 μ 2.9	6 μ 2.9	52 μ 3.7	33 μ 5.4	29 μ 5.0
Resnet101	71 -10.6	41 μ 22.6	32 μ 21.6	50 μ 12.4	23 μ 13.7	18 μ 12.3	99 μ 0.8	100 μ 0.08	100 μ 0.04	49 μ 2.9	17 μ 3.4	8 μ 2.1	11 μ 1.9	1 μ 0.4	1 μ 0.3	19 μ 2.5	7 μ 2.4	5 μ 2.3	50 μ 3.6	31 μ 5.0	27 μ 4.4
Resnet101+CW	72 μ 10.0	40 μ 20.2	31 μ 19.9	53 μ 9.6	25 μ 15.0	20 μ 14.0	99 μ 0.1	100 μ 0.13	100 μ 0.08	49 μ 2.7	16 μ 3.9	8 μ 2.6	9 μ 1.5	1 μ 0.3	1 μ 0.2	21 μ 2.4	7 μ 2.1	6 μ 1.9	51 μ 3.2	32 μ 5.4	27 μ 5.2

Appendix L

Density plots for number of pixels per class over images

Figure L.1: Here each image from (a) to (t) displays the distribution of number of pixels for each class separately over all images in the dataset.



Appendix M

Discovered best Cityscapes model visualisation

The visualisation as well as the saved state of the model are present in the submitted supplementary materials.

References

- Abdelfattah, M. S., Mehrotra, A., Dudziak, Ł. and Lane, N. D., 2021. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*.
- Abidi, B., Zheng, Y., Gribok, A. and Abidi, M., 2005. Screener evaluation of pseudo-colored single energy x-ray luggage images. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, IEEE, 35–35.
- Abidi, B. R., Zheng, Y., Gribok, A. V. and Abidi, M. A., 2006. Improving weapon detection in single energy x-ray images through pseudocoloring. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36 (6), 784–796.
- Agarwal, S., Kumar, B. and Singh, D., 2015. Non-invasive concealed weapon detection and identification using v band millimeter wave imaging radar system. *2015 National Conference on Recent Advances in Electronics & Computer Engineering (RAECE)*, IEEE, 258–262.
- Aggarwal, S., Singh, N. and Mishra, K., 2021. Digital image analysis is a silver bullet to covid-19 pandemic. *Computational Intelligence Methods in COVID-19: Surveillance, Prevention, Prediction and Diagnosis*, Springer, 397–414.
- Agyemang, D. B. and Bader, M., 2019. Surface crack detection using hierarchal convolutional neural network. *UK Workshop on Computational Intelligence*, Springer, 173–186.
- Ahmed, W. S. et al., 2020. The impact of filter size and number of filters on classification accuracy in cnn. *2020 International conference on computer science and software engineering (CSASE)*, IEEE, 88–93.
- Aichert, A., Wiczorek, M., Wang, J., Kreiser, M., Wang, L., Fallavollita, P. and Navab, N., 2012. The colored x-rays. *Workshop on Augmented Environments for Computer-Assisted Interventions*, Springer, 45–54.
- Akçay, S., Atapour-Abarghouei, A. and Breckon, T. P., 2018. Ganomaly: Semi-supervised anomaly detection via adversarial training. *Asian Conference on Computer Vision*, Springer, 622–637.

- Akcay, S. and Breckon, T., 2020. Towards automatic threat detection: A survey of advances of deep learning within x-ray security imaging. *arXiv preprint arXiv:2001.01293*.
- Albawi, S., Mohammed, T. A. and Al-Zawi, S., 2017. Understanding of a convolutional neural network. *2017 international conference on engineering and technology (ICET)*, IEEE, 1–6.
- Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., Awwal, A. A. S. and Asari, V. K., 2018. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.
- Alonso, I., Riazuelo, L. and Murillo, A. C., 2020. Mininet: An efficient semantic segmentation convnet for real-time robotic applications. *IEEE Transactions on Robotics*, 36 (4), 1340–1347.
- Alqahtani, A., Xie, X., Jones, M. W. and Essa, E., 2021. Pruning cnn filters via quantifying the importance of deep visual representations. *Computer Vision and Image Understanding*, 208, 103220.
- Alsabhan, W., Alotaiby, T. et al., 2022. Automatic building extraction on satellite images using unet and resnet50. *Computational Intelligence and Neuroscience*, 2022.
- Alvernaz, S. and Togelius, J., 2017. Autoencoder-augmented neuroevolution for visual doom playing. *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, 1–8.
- Aly, A., Weikersdorfer, D. and Delaunay, C., 2019. Optimizing deep neural networks with multiple search neuroevolution. *arXiv preprint arXiv:1901.05988*.
- An, J., Zhang, H., Zhu, Y. and Yang, J., 2019. Semantic segmentation for prohibited items in baggage inspection. *International Conference on Intelligent Science and Big Data Engineering*, Springer, 495–505.
- Anand, A., Pugalenth, G., Fogel, G. B. and Suganthan, P., 2010. An approach for classification of highly imbalanced data using weighting and undersampling. *Amino acids*, 39 (5), 1385–1391.
- Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y. et al., 2017. A closer look at memorization in deep networks. *International conference on machine learning*, PMLR, 233–242.
- Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. et al., 2020. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58, 82–115.

- Assunção, F., Lourenço, N., Machado, P. and Ribeiro, B., 2019. Fast denser: Efficient deep neuroevolution. *European Conference on Genetic Programming*, Springer, 197–212.
- Azarang, A., Manoochehri, H. E. and Kehtarnavaz, N., 2019. Convolutional autoencoder-based multispectral image fusion. *IEEE Access*, 7, 35673–35683.
- Babenko, A. and Lempitsky, V., 2016. Efficient indexing of billion-scale datasets of deep descriptors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2055–2063.
- Back, T., Hammel, U. and Schwefel, H.-P., 1997. Evolutionary computation: Comments on the history and current state. *IEEE transactions on Evolutionary Computation*, 1 (1), 3–17.
- Bagheri-Esfeh, H., Rostamzadeh-Renani, M., Rostamzadeh-Renani, R. and Safihkani, H., 2020. Multi-objective optimisation of drag and lift coefficients of a car integrated with canards. *International Journal of Computational Fluid Dynamics*, 34 (5), 346–362.
- Balcioglu, A., Gillani, R., Doron, M., Burnell, K., Ku, T., Erisir, A., Chung, K., Segev, I. and Nedivi, E., 2023. Mapping thalamic innervation to individual I2/3 pyramidal neurons and modeling their ‘readout’ of visual input. *Nature Neuroscience*, 26 (3), 470–480.
- Baldi, P., 2012. Autoencoders, unsupervised learning, and deep architectures. *Proceedings of ICML workshop on unsupervised and transfer learning*, JMLR Workshop and Conference Proceedings, 37–49.
- Bandi, A., Adapa, P. V. S. R. and Kuchi, Y. E. V. P. K., 2023. The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15 (8), 260.
- Barbiero, P., Squillero, G. and Tonda, A., 2019. Beyond coreset discovery: evolutionary archetypes. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 47–48.
- Barbiero, P., Squillero, G. and Tonda, A., 2020. Uncovering coresets for classification with multi-objective evolutionary algorithms. *arXiv preprint arXiv:2002.08645*.
- Barkan, O., Caciularu, A., Katz, O. and Koenigstein, N., 2020. Attentive item2vec: Neural attentive user representations. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 3377–3381.
- Bartley, W. A. and Cohen, M. A., 1998. The effect of concealed weapons laws: An extreme bound analysis. *Economic Inquiry*, 36 (2), 258–265.

- Barua, S., Islam, M. M., Yao, X. and Murase, K., 2012. Mwmote—majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on knowledge and data engineering*, 26 (2), 405–425.
- Bassen, H., Mendoza, G., Razjouyan, A., Eslami, M. and Shahshahan, N., 2019. Electromagnetic radiation detection apparatus and method of detecting low levels of millimeter wave electromagnetic radiation. US Patent App. 10/267,837.
- Bayes, T., 1763. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53), 370–418.
- Bender, G., Liu, H., Chen, B., Chu, G., Cheng, S., Kindermans, P.-J. and Le, Q. V., 2020. Can weight sharing outperform random architecture search? an investigation with tunas. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14323–14332.
- Bengio, Y., Courville, A. and Vincent, P., 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8), 1798–1828.
- Bengio, Y., Paiement, J.-f., Vincent, P., Delalleau, O., Roux, N. and Ouimet, M., 2003. Out-of-sampl, isomap, mds, eigenmaps, and spectral clustering. *Advances in neural information processing systems*, 16, 177–184.
- Bermeitinger, B., Hrycej, T. and Handschuh, S., 2019. Representational capacity of deep neural networks—a computing study. *arXiv preprint arXiv:1907.08475*.
- Bertels, J., Eelbode, T., Berman, M., Vandermeulen, D., Maes, F., Bisschops, R. and Blaschko, M. B., 2019. Optimizing the dice score and jaccard index for medical image segmentation: Theory and practice. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 92–100.
- Bhalgat, Y., Shah, M. and Awate, S., 2018. Annotation-cost minimization for medical image segmentation using suggestive mixed supervision fully convolutional networks. *arXiv preprint arXiv:1812.11302*.
- Bhargava, M., Chen, C.-C., Ryoo, M. S. and Aggarwal, J. K., 2007. Detection of abandoned objects in crowded environments. *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, IEEE, 271–276.
- Bisong, E., 2019. Google automl: cloud vision. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, 581–598.

- Björnson, E., Jorswieck, E. et al., 2013. Optimal resource allocation in coordinated multi-cell systems. *Foundations and Trends® in Communications and Information Theory*, 9 (2–3), 113–381.
- Blalock, D., Ortiz, J. J. G., Frankle, J. and Gutttag, J., 2020. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*.
- Blank, J. and Deb, K., 2020. pymoo: Multi-objective optimization in python. *IEEE Access*, 8, 89497–89509.
- Blickle, T. and Thiele, L., 1996. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4 (4), 361–394.
- Bolz Jr, F., Dudonis, K. J. and Schulz, D. P., 2016. *The counterterrorism handbook: Tactics, procedures, and techniques*. Crc Press.
- Bonet, D., Ortega, A., Ruiz-Hidalgo, J. and Shekkizhar, S., 2021. Channel-wise early stopping without a validation set via nnk polytope interpolation. *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, 351–358.
- Borkowski, A. A., Wilson, C. P., Borkowski, S. A., Thomas, L. B., Deland, L. A., Grewe, S. J. and Mastorides, S. M., 2019. Google auto ml versus apple create ml for histopathologic cancer diagnosis; which algorithms are better? *arXiv preprint arXiv:1903.08057*.
- Buxhoeveden, D. P. and Casanova, M. F., 2002. The minicolumn hypothesis in neuroscience. *Brain*, 125 (5), 935–951.
- Campbell, T. and Broderick, T., 2018. Bayesian coreset construction via greedy iterative geodesic ascent. *International Conference on Machine Learning*, PMLR, 698–706.
- Castilla, T., Martínez, M. S., Leguía, M., Larrabide, I. and Orlando, J. I., 2022. A resnet is all you need? modeling a strong baseline for detecting referable diabetic retinopathy in fundus images. *arXiv preprint arXiv:2210.03180*.
- Censor, Y., 1977. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4 (1), 41–59.
- Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41 (3), 1–58.
- Chapelle, O., Haffner, P. and Vapnik, V. N., 1999. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10 (5), 1055–1064.

- Charte, F., Rivera, A. J., Martínez, F. and del Jesus, M. J., 2020. Evoaaa: An evolutionary methodology for automated neural autoencoder architecture search. *Integrated Computer-Aided Engineering*, (Preprint), 1–21.
- Chavallaz, A., Schwaninger, A., Michel, S. and Sauer, J., 2019. Expertise, automation and trust in x-ray screening of cabin baggage. *Frontiers in psychology*, 10, 256.
- 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*, 16, 321–357.
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A. L., 2014. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.
- Chen, L.-C., Papandreou, G., Schroff, F. and Adam, H., 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.
- Chen, T., Kornblith, S., Norouzi, M. and Hinton, G., 2020. A simple framework for contrastive learning of visual representations. *International conference on machine learning*, PMLR, 1597–1607.
- Chen, Z., Fu, Y., Wang, Y.-X., Ma, L., Liu, W. and Hebert, M., 2019. Image deformation meta-networks for one-shot learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8680–8689.
- Cheng, B., Girshick, R., Dollár, P., Berg, A. C. and Kirillov, A., 2021. Boundary iou: Improving object-centric image segmentation evaluation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15334–15342.
- Cheng, Z., Sun, H., Takeuchi, M. and Katto, J., 2018. Performance comparison of convolutional autoencoders, generative adversarial networks and super-resolution for image compression. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2613–2616.
- Cherepennikov, Y. M., Rezaev, R. O., Stuchebrov, S. G. and Wagner, A. R., 2015. Dual-wave x-ray absorptiometry in multiphase flow metering. *RREPS-15. Radiation from Relativistic Electrons in Periodic Structures: XI International Symposium, 6-11 September 2015, Saint Petersburg, Russia.—Tomsk, 2015.*, 43.
- Chouhan, V., Singh, S. K., Khamparia, A., Gupta, D., Tiwari, P., Moreira, C., Damaševičius, R. and De Albuquerque, V. H. C., 2020. A novel transfer learning based approach for pneumonia detection in chest x-ray images. *Applied Sciences*, 10 (2), 559.

- Chrabaszcz, P., Loshchilov, I. and Hutter, F., 2017. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*.
- Chu, X., Zhang, B. and Xu, R., 2021. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12239–12248.
- Clarkson, K. L., 2010. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6 (4), 1–30.
- Cochocki, A. and Unbehauen, R., 1993. *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc.
- Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A. et al., 2007. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer.
- Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B., 2015. The cityscapes dataset. *CVPR Workshop on the Future of Datasets in Vision*, volume 2.
- Cunningham, P., Cord, M. and Delany, S. J., 2008. Supervised learning. *Machine learning techniques for multimedia*, Springer, 21–49.
- Cyganek, B., 2013. *Object detection and recognition in digital images: theory and practice*. John Wiley & Sons.
- D'AGOSTINO, R. and Pearson, E. S., 1973. Tests for departure from normality. empirical results for the distributions of b^2 and square b . *Biometrika*, 60 (3), 613–622.
- d'Agostino, R. B., 1971. An omnibus test of normality for moderate and large size samples. *Biometrika*, 58 (2), 341–348.
- Dai, J., Lu, Y. and Wu, Y.-N., 2014. Generative modeling of convolutional neural networks. *arXiv preprint arXiv:1412.6296*.
- Dai, W., Dai, C., Qu, S., Li, J. and Das, S., 2017. Very deep convolutional neural networks for raw waveforms. *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 421–425.
- Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. *international Conference on computer vision & Pattern Recognition (CVPR'05)*, IEEE Computer Society, volume 1, 886–893.
- Darken, C., Chang, J., Moody, J. et al., 1992. Learning rate schedules for faster stochastic gradient search. *Neural networks for signal processing*, Citeseer, volume 2.

- Davison, J., 2017. Devol - deep neural network evolution. URL <https://github.com/joeddav/devol>.
- De Falco, I., Della Cioppa, A. and Tarantino, E., 2002. Mutation-based genetic algorithm: performance evaluation. *Applied Soft Computing*, 1 (4), 285–299.
- Deb, K., 2001a. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons.
- Deb, K., 2001b. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons.
- Deb, K., 2014. Multi-objective optimization. *Search methodologies*, Springer, 403–449.
- Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., 2002a. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6 (2), 182–197.
- Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., 2002b. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6 (2), 182–197.
- Deb, K., Sindhya, K. and Okabe, T., 2007. Self-adaptive simulated binary crossover for real-parameter optimization. *Proceedings of the 9th annual conference on genetic and evolutionary computation*, 1187–1194.
- Deb, K. and Tiwari, S., 2008. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185 (3), 1062–1087.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, IEEE, 248–255.
- Deng, L., 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29 (6), 141–142.
- Deng, S., Wang, C., Li, Z., Zhang, N., Dai, Z., Chen, H., Xiong, F., Yan, M., Chen, Q., Chen, M. et al., 2023a. Construction and applications of billion-scale pre-trained multimodal business knowledge graph. *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, IEEE, 2988–3002.
- Deng, Y., Yang, Y., Mirzasoleiman, B. and Gu, Q., 2023b. Robust learning with progressive data expansion against spurious correlation. *arXiv preprint arXiv:2306.04949*.

- Department of Homeland Security, 2017. Passenger screening algorithm challenge. URL <https://www.kaggle.com/c/passenger-screening-algorithm-challenge/overview/timeline>.
- Derrac, J., García, S. and Herrera, F., 2012. A survey on evolutionary instance selection and generation. *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*, IGI Global, 233–266.
- Dhamija, T., Gupta, A., Gupta, S., Anjum, Katarya, R. and Singh, G., 2023. Semantic segmentation in medical images through transfused convolution and transformer networks. *Applied Intelligence*, 53 (1), 1132–1148.
- Dhanya, K., Vajipayajula, S., Srinivasan, K., Tibrewal, A., Kumar, T. S. and Kumar, T. G., 2023. Detection of network attacks using machine learning and deep learning models. *Procedia Computer Science*, 218, 57–66.
- Dimanov, B., 2021. *Interpretable Deep Learning: Beyond Feature-Importance with Concept-based Explanations*. Ph.D. thesis, University of Cambridge.
- Dimanov, D., 2019. *Neuroevolution for Weapons Detection in Digital Images*. Master's thesis, Bournemouth University.
- Dimanov, D., Balaguer-Ballester, E., Singleton, C. and Rostami, S., 2021. Moncae: Multi-objective neuroevolution of convolutional autoencoders. *arXiv preprint arXiv:2106.11914*.
- Dimanov, D. and Rostami, S., 2019. Kosi-key object detection for sentiment insights. *UK Workshop on Computational Intelligence*, Springer, 296–306.
- Dimanov, D., Sigleton, C., Rostami, S. and Ballaguer-Ballester, E., 2022. Ramoss- resource aware neuroevolution for multi-objective semantic segmentation. *UK Workshop on Computational Intelligence*, Springer, 428–438.
- Doke, A. and Gaikwad, M., 2021. Survey on automated machine learning (automl) and meta learning. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, 1–5.
- Domingo-Perez, F., Lazaro-Galilea, J. L., Wieser, A., Martin-Gorostiza, E., Salido-Monzu, D. and de la Llana, A., 2016. Sensor placement determination for range-difference positioning using evolutionary multi-objective optimization. *Expert Systems with Applications*, 47, 95–105.
- Došilović, F. K., Brčić, M. and Hlupić, N., 2018. Explainable artificial intelligence: A survey. *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, IEEE, 0210–0215.

- Dosovitskiy, A. and Brox, T., 2016. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*.
- Duan, S., Pan, W., Leng, Y. and Zhang, X., 2023. Two resnet mini architectures for aircraft wake vortex identification. *IEEE Access*, 11, 20515–20523.
- Dubey, A., Chatterjee, M. and Ahuja, N., 2018. Coreset-based neural network compression. *Proceedings of the European Conference on Computer Vision (ECCV)*, 454–470.
- Dumagpi, J. K. and Jeong, Y.-J., 2020. Evaluating gan-based image augmentation for threat detection in large-scale xray security images. *Applied Sciences*, 11 (1), 36.
- Dürr, P., Mattiussi, C. and Floreano, D., 2006. Neuroevolution with analog genetic encoding. *Parallel Problem Solving from Nature-PPSN IX*, Springer, 671–680.
- Dutta, H., Kwon, K. H. and Rao, H. R., 2018. A system for intergroup prejudice detection: The case of microblogging under terrorist attacks. *Decision Support Systems*, 113, 11–21.
- Eaton, M., 2015. Evolutionary algorithms and the control of systems. *Evolutionary Humanoid Robotics*, Springer, 9–20.
- Ellefsen, K. O., Lepikson, H. A. and Albiez, J. C., 2017. Multiobjective coverage path planning: Enabling automated inspection of complex, real-world structures. *Applied Soft Computing*, 61, 264–282.
- Elsken, T., Metzen, J. H. and Hutter, F., 2018a. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*.
- Elsken, T., Metzen, J. H. and Hutter, F., 2018b. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.
- Elsken, T., Metzen, J. H. and Hutter, F., 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20 (1), 1997–2017.
- Emam, Z. A. S., Chu, H.-M., Chiang, P.-Y., Czaja, W., Leapman, R., Goldblum, M. and Goldstein, T., 2021. Active learning at the imagenet scale. *arXiv preprint arXiv:2111.12880*.
- Emuoyibofarhe, J. O., Ajisafe, D., Babatunde, R. S. and Christoph, M., 2020. Early skin cancer detection using deep convolutional neural networks on mobile smartphone. *International Journal of Information Engineering & Electronic Business*, 12 (2).
- Feder, R., Spiller, E., Topalian, J., Broers, A., Gudat, W., Panessa, B., Zadunaisky, Z. and Sedat, J., 1977. High-resolution soft x-ray microscopy. *Science*, 197 (4300), 259–260.

- Fellbaum, C., 2010. Wordnet. *Theory and applications of ontology: computer applications*, Springer, 231–243.
- Fenster, A. and Chiu, B., 2006. Evaluation of segmentation algorithms for medical imaging. *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, IEEE, 7186–7189.
- Floreano, D., Dürr, P. and Mattiussi, C., 2008. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1 (1), 47–62.
- Fogel, D. B., 2000. Introduction to evolutionary computation. *Evolutionary computation*, 1.
- Forsyth, D. A. and Ponce, J., 2002. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.
- Frankle, J. and Carbin, M., 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.
- Freilich, J. D., Chermak, S. M., Connell, N. M., Klein, B. R. and Greene-Colozzi, E. A., 2022. Using open-source data to better understand and respond to american school shootings: introducing and exploring the american school shooting study (tasss). *Journal of school violence*, 21 (2), 93–118.
- Fukushima, K. and Miyake, S., 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. *Competition and cooperation in neural nets*, Springer, 267–285.
- Gallego, G., Cuevas, C., Mohedano, R. and Garcia, N., 2013. On the mahalanobis distance classification criterion for multidimensional normal distributions. *IEEE Transactions on Signal Processing*, 61 (17), 4387–4396.
- Galván, E. and Mooney, P., 2021. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Transactions on Artificial Intelligence*.
- Gaus, Y. F. A., Bhowmik, N., Akçay, S., Guillen-Garcia, P. M., Barker, J. W. and Breckon, T. P., 2019. Evaluation of a dual convolutional neural network architecture for object-wise anomaly detection in cluttered x-ray security imagery. *arXiv preprint arXiv:1904.05304*.
- Geiger, A., Lenz, P., Stiller, C. and Urtasun, R., 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32 (11), 1231–1237.
- Gelana, F. and Yadav, A., 2019. Firearm detection from surveillance cameras using image processing and machine learning techniques. *Smart Innovations in Communication and Computational Sciences*, Springer, 25–34.

- Gervais, D., 2019. Exploring the interfaces between big data and intellectual property law. *J. Intell. Prop. Info. Tech. & Elec. Com. L.*, 10, 3.
- Ghadakchi, V., Khodadadi, A. and Termehchy, A., 2019. Less data delivers higher search effectiveness for keyword queries. *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*, 109–120.
- Gil, Y., Oh, Y., Cho, M. and Namkung, W., 2011. Radiography simulation on single-shot dual-spectrum x-ray for cargo inspection system. *Applied Radiation and Isotopes*, 69 (2), 389–393.
- Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K. V., Joulin, A. and Misra, I., 2023. Imagebind: One embedding space to bind them all. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15180–15190.
- Girshick, R., 2015. Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*, 1440–1448.
- Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.
- Givnan, S., Chalmers, C., Fergus, P., Ortega-Martorell, S. and Whalley, T., 2022. Anomaly detection using autoencoder reconstruction upon industrial motors. *Sensors*, 22 (9), 3166.
- Goenka, A. and Sitara, K., 2022. Weapon detection from surveillance images using deep learning. *2022 3rd International Conference for Emerging Technology (INCET)*, IEEE, 1–6.
- Goldberg, D. E. and Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, Elsevier, volume 1, 69–93.
- Goldstein, J. A., Nateghi, R., Irmakci, I. and Cooper, L. A., 2023. Machine learning classification of placental villous infarction, perivillous fibrin deposition, and intervillous thrombus. *Placenta*, 135, 43–50.
- Gondara, L., 2016. Medical image denoising using convolutional denoising autoencoders. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, IEEE, 241–246.
- Goodfellow, I., Bengio, Y. and Courville, A., 2016a. *Deep learning*. MIT press.
- Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., 2016b. *Deep learning*, volume 1. MIT press Cambridge.

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., 2672–2680. URL <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.
- Grega, M., Mاتیolański, A., Guzik, P. and Leszczuk, M., 2016. Automated detection of firearms and knives in a cctv image. *Sensors*, 16 (1), 47.
- Gruau, F., 1994a. Automatic definition of modular neural networks. *Adaptive behavior*, 3 (2), 151–183.
- Gruau, F., 1994b. Neural network synthesis using cellular encoding and the genetic algorithm.
- Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. et al., 2018. Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354–377.
- Guerreiro, A. P., Fonseca, C. M. and Paquete, L., 2020. The hypervolume indicator: Problems and algorithms. *arXiv preprint arXiv:2005.00515*.
- Guha, R., Ao, W., Kelly, S., Boddeti, V., Goodman, E., Banzhaf, W. and Deb, K., 2023. Moaz: A multi-objective automl-zero framework. *Proceedings of the Genetic and Evolutionary Computation Conference*, 485–492.
- Guidotti, R., Monreale, A., Matwin, S. and Pedreschi, D., 2020. Black box explanation by learning image exemplars in the latent feature space. *arXiv preprint arXiv:2002.03746*.
- Gunasekaran, K. P., 2023. Ultra sharp: Study of single image super resolution using residual dense network.
- Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S. and Yang, G.-Z., 2019. Xai—explainable artificial intelligence. *Science robotics*, 4 (37), eaay7120.
- Guo, C., Zhao, B. and Bai, Y., 2022. Deepcore: A comprehensive library for coreset selection in deep learning. *arXiv preprint arXiv:2204.08499*.
- Guo, Y., Li, Y., Wang, L. and Rosing, T., 2019a. Depthwise convolution is all you need for learning multiple visual domains. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 8368–8375.
- Guo, Y., Zheng, Y., Tan, M., Chen, Q., Chen, J., Zhao, P. and Huang, J., 2019b. Nat: Neural architecture transformer for accurate and compact architectures. *arXiv preprint arXiv:1910.14488*.

- Hadjiivanov, A. and Blair, A., 2016. Complexity-based speciation and genotype representation for neuroevolution. *2016 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 3092–3101.
- Hamaguchi, R., Fujita, A., Nemoto, K., Imaizumi, T. and Hikosaka, S., 2018. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. *2018 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 1442–1450.
- Har-Peled, S. and Kushal, A., 2005. Smaller coresets for k-median and k-means clustering. *Proceedings of the twenty-first annual symposium on Computational geometry*, 126–134.
- Har-Peled, S. and Mazumdar, S., 2004. On coresets for k-means and k-median clustering. *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 291–300.
- Haralick, R. M., Shanmugam, K. and Dinstein, I. H., 1973. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 6, 610–621.
- Haralick, R. M. and Shapiro, L. G., 1985. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29 (1), 100–132.
- Harzallah, H., Jurie, F. and Schmid, C., 2009. Combining efficient object localization and image classification. *2009 IEEE 12th international conference on computer vision*, IEEE, 237–244.
- Hassan, T., Shafay, M., Akçay, S., Khan, S., Bennamoun, M., Damiani, E. and Werghi, N., 2020. Meta-transfer learning driven tensor-shot detector for the autonomous localization and recognition of concealed baggage threats. *Sensors*, 20 (22), 6450.
- Hayler, W. P., Rutherford, M., Jones, M. A., Kirk, A. D., Gilbert Walter Vanorder, I., Hudson, R. D., Mason, P. and Termini, J., 2019. X-ray scanning system and method. US Patent 10,203,426.
- He, K., Fan, H., Wu, Y., Xie, S. and Girshick, R., 2020. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.
- He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961–2969.
- He, K. and Sun, J., 2015. Convolutional neural networks at constrained time cost. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5353–5360.

- He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385. URL <http://arxiv.org/abs/1512.03385>.
- He, K., Zhang, X., Ren, S. and Sun, J., 2016a. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- He, K., Zhang, X., Ren, S. and Sun, J., 2016b. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, 770–778.
- He, X., Zhao, K. and Chu, X., 2021. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212, 106622.
- Henzler, P., Rasche, V., Ropinski, T. and Ritschel, T., 2018. Single-image tomography: 3d volumes from 2d cranial x-rays. *Computer Graphics Forum*, Wiley Online Library, volume 37, 377–388.
- Hinton, G., 2014. What is wrong with convolutional neural nets? *Conference on Neural Information Processing Systems*. URL <https://www.youtube.com/watch?v=rTawFwUvnLE>.
- Hinton, G., Vinyals, O. and Dean, J., 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Hinton, G. E., 2012. A practical guide to training restricted boltzmann machines. *Neural networks: Tricks of the trade*, Springer, 599–619.
- Hinton, G. E., Sabour, S. and Frosst, N., 2018. Matrix capsules with em routing. *International Conference on Learning Representations*.
- Ho, T. K. and Basu, M., 2002. Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, 24 (3), 289–300.
- Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2 (5), 359–366.
- Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. et al., 2019. Searching for mobilenetv3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314–1324.
- Hsu, C.-H., Chang, S.-H., Liang, J.-H., Chou, H.-P., Liu, C.-H., Chang, S.-C., Pan, J.-Y., Chen, Y.-T., Wei, W. and Juan, D.-C., 2018. Monas: Multi-objective neural architecture search using reinforcement learning. *arXiv preprint arXiv:1806.10332*.

- Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q., 2017. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.
- Huang, G., Sun, Y., Liu, Z., Sedra, D. and Weinberger, K. Q., 2016. Deep networks with stochastic depth. *European conference on computer vision*, Springer, 646–661.
- Huang, G.-B., Chen, L., Siew, C. K. et al., 2006. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, 17 (4), 879–892.
- Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., Han, X., Chen, Y.-W. and Wu, J., 2020. Unet 3+: A full-scale connected unet for medical image segmentation. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 1055–1059.
- Huang, T., 1996. Computer vision: Evolution and promise. *Cern*.
- Husbands, P., Harvey, I., Cliff, D. and Miller, G., 1994. The use of genetic algorithms for the development of sensorimotor control systems. *Proceedings of PerAc'94. From Perception to Action*, IEEE, 110–121.
- Hussain, M., Bird, J. J. and Faria, D. R., 2018. A study on cnn transfer learning for image classification. *UK Workshop on computational Intelligence*, Springer, 191–202.
- Hussein, N. J., Hu, F., Hu, H. and Rahem, A. T., 2016. IR and Multi Scale Retinex image enhancement for concealed weapon detection. *Indonesian Journal of Electrical Engineering and Computer Science*, 1 (2), 399–405.
- Hussin, R., Juhari, M. R., Kang, N. W., Ismail, R. and Kamarudin, A., 2012. Digital image processing techniques for object detection from complex background image. *Procedia Engineering*, 41, 340–344.
- Hutter, F., Kotthoff, L. and Vanschoren, J., 2019a. *Automated machine learning: methods, systems, challenges*. Springer Nature.
- Hutter, F., Kotthoff, L. and Vanschoren, J., 2019b. Automated machine learning-methods, systems, challenges.
- Hwang, C.-L. and Masud, A. S. M., 2012. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media.
- Isaac-Medina, B. K., Willcocks, C. G. and Breckon, T. P., 2021. Multi-view object detection using epipolar constraints within cluttered x-ray security imagery. *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 9889–9896.

- Ishibuchi, H., Setoguchi, Y., Masuda, H. and Nojima, Y., 2016. Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21 (2), 169–190.
- Jankowski, N. and Grochowski, M., 2004. Comparison of instances selection algorithms i. algorithms survey. *International conference on artificial intelligence and soft computing*, Springer, 598–603.
- Jansen, T., Wegener, I. et al., 2002. The analysis of evolutionary algorithms—a proof that crossover really can help. *Algorithmica*, 34 (1), 47–66.
- Jenkins, W., 2006. Neural network weight training by mutation. *Computers & structures*, 84 (31-32), 2107–2112.
- Jha, A., Peterson, J. C. and Griffiths, T. L., 2023. Extracting low-dimensional psychological representations from convolutional neural networks. *Cognitive science*, 47 (1), e13226.
- Jiang, Y., Zhu, X., Wang, X., Yang, S., Li, W., Wang, H., Fu, P. and Luo, Z., 2018. R 2 cnn: Rotational region cnn for arbitrarily-oriented scene text detection. *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 3610–3615.
- Jin, X., Chi, J., Peng, S., Tian, Y., Ye, C. and Li, X., 2016. Deep image aesthetics classification using inception modules and fine-tuning connected layer. *2016 8th International Conference on Wireless Communications & Signal Processing (WCSP)*, IEEE, 1–6.
- Ju, J., Jung, H., Oh, Y. and Kim, J., 2022. Extending contrastive learning to unsupervised coreset selection. *IEEE Access*, 10, 7704–7715.
- Juefei-Xu, F., Naresh Boddeti, V. and Savvides, M., 2017. Local binary convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 19–28.
- Kachitvichyanukul, V., 2012. Comparison of three evolutionary algorithms: Ga, pso, and de. *Industrial Engineering & Management Systems*, 11 (3), 215–223.
- Kakani, V., Nguyen, V. H., Kumar, B. P., Kim, H. and Pasupuleti, V. R., 2020. A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, 2, 100033.
- Kamble, K., Sontakke, S., Mundada, P. and Pawar, A., 2020. Threat detection with facial expression and suspicious weapon. *Applied Computer Vision and Image Processing*, Springer, 386–397.
- Kang, J.-S., Kang, J., Kim, J.-J., Jeon, K.-W., Chung, H.-J. and Park, B.-H., 2023. Neural architecture search survey: A computer vision perspective. *Sensors*, 23 (3), 1713.

- Kang, L.-W., Lin, C.-W. and Fu, Y.-H., 2011. Automatic single-image-based rain streaks removal via image decomposition. *IEEE transactions on image processing*, 21 (4), 1742–1755.
- Karmaker, S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C. and Veeramachaneni, K., 2021. Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)*, 54 (8), 1–36.
- Kazhdan, D., Dimanov, B., Jamnik, M., Liò, P. and Weller, A., 2020. Now you see me (cme): concept-based model extraction. *arXiv preprint arXiv:2010.13233*.
- Kazhdan, D., Dimanov, B., Terre, H. A., Jamnik, M., Liò, P. and Weller, A., 2021. Is disentanglement all you need? comparing concept-based & disentanglement approaches. *arXiv preprint arXiv:2104.06917*.
- Kelly, S., Park, D. S., Song, X., McIntire, M., Nashikkar, P., Guha, R., Banzhaf, W., Deb, K., Boddeti, V. N., Tan, J. et al., 2023. Discovering adaptable symbolic algorithms from scratch. *arXiv preprint arXiv:2307.16890*.
- Khan, M. Z., 2023. *A Novel Deep Learning-Based Framework for Context-Aware Semantic Segmentation in Medical Imaging*. Ph.D. thesis, University of Missouri-Kansas City.
- Khan, S., Rahmani, H., Shah, S. A. A. and Bennamoun, M., 2018. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8 (1), 1–207.
- Khairat, S., Feyzmahdavian, H. R. and Johansson, M., 2017. Mini-batch gradient descent: Faster convergence under data sparsity. *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2880–2887.
- Kilkenny, M. F. and Robinson, K. M., 2018. Data quality: “garbage in–garbage out”.
- Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G. and Iyer, R., 2021. Glisten: Generalization based data subset selection for efficient and robust learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8110–8118.
- Kim, J., Skalenda, W. F., Tomich, J. L. and Samaniego, R., 2018. Radar detection of a concealed object on a body. US Patent 9,903,948.
- Kitano, H., 1990. Designing neural networks using genetic algorithms with graph generation system. *Complex systems*, 4, 461–476.
- Koenig, A. C., 2002. A study of mutation methods for evolutionary algorithms. *University of Missouri-Rolla*.

- Krizhevsky, A., 2009. Learning multiple layers of features from tiny images. Technical report.
- Krizhevsky, A., Nair, V. and Hinton, G., 2014. The cifar-10 dataset. *online: <http://www.cs.toronto.edu/kriz/cifar.html>*, 55, 5.
- Krizhevsky, A., Sutskever, I. and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097–1105.
- Kumar, S. and Bentley, P. J., 2003. *On growth, form and computers*. Elsevier.
- Lacoste, A., Luccioni, A., Schmidt, V. and Dandres, T., 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.
- Lang, K. J., Waibel, A. H. and Hinton, G. E., 1990. A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3 (1), 23–43.
- Langlois, F., Rhumorbarbe, D., Werner, D., Florquin, N., Caneppele, S. and Rossy, Q., 2022. International weapons trafficking from the united states of america: a crime script analysis of the means of transportation. *Global Crime*, 1–22.
- Larranaga, P., Karshenas, H., Bielza, C. and Santana, R., 2013. A review on evolutionary algorithms in bayesian network learning and inference tasks. *Information Sciences*, 233, 109–125.
- Lawrence, S., Giles, C. L., Tsoi, A. C. and Back, A. D., 1997. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8 (1), 98–113.
- Lebedev, V. and Lempitsky, V., 2018. Speeding-up convolutional neural networks: A survey. *Bulletin of the Polish Academy of Sciences. Technical Sciences*, 66 (6).
- LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. et al., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11), 2278–2324.
- LeCun, Y., Cortes, C. and Burges, C., 1999. The mnist dataset of handwritten digits (images). *NYU: New York, NY, USA*.
- Lee, I., 2017. Big data: Dimensions, evolution, impacts, and challenges. *Business Horizons*, 60 (3), 293–303.
- Lee, S. and Song, B. C., 2023. Fast filter pruning via coarse-to-fine neural architecture search and contrastive knowledge transfer. *IEEE Transactions on Neural Networks and Learning Systems*.
- Lee, Y., Choi, T. J. and Ahn, C. W., 2019. Multi-objective evolutionary approach to select security solutions. *CAAI Transactions on Intelligence Technology*, 2 (2), 64–67.

- Lehman, J. and Miikkulainen, R., 2013. Neuroevolution. *Scholarpedia*, 8 (6), 30977.
- Lei, X., Pan, H. and Huang, X., 2019. A dilated cnn model for image classification. *IEEE Access*, 7, 124087–124095.
- Li, K., Min, M. R. and Fu, Y., 2019. Rethinking zero-shot learning: A conditional visual classification perspective. *Proceedings of the IEEE International Conference on Computer Vision*, 3583–3592.
- Li, S., Sun, L. and Li, Q., 2023. Clip-reid: Exploiting vision-language model for image re-identification without concrete text labels. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 1405–1413.
- Li, S. and Wu, S., 2022. Low-cost millimeter wave frequency scanning based synthesis aperture imaging system for concealed weapon detection. *IEEE Transactions on Microwave Theory and Techniques*.
- Li, Y., Wang, Z., Xie, Y., Ding, B., Zeng, K. and Zhang, C., 2021. Automl: From methodology to application. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 4853–4856.
- Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. and Sun, J., 2018. Detnet: Design backbone for object detection. *Proceedings of the European conference on computer vision (ECCV)*, 334–350.
- Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K. and Li, Z., 2019a. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*.
- Liang, K. J., Sigman, J. B., Spell, G. P., Strellis, D., Chang, W., Liu, F., Mehta, T. and Carin, L., 2019b. Toward automatic threat recognition for airport x-ray baggage screening with deep convolutional object detection. *arXiv preprint arXiv:1912.06329*.
- Liashchynskiy, P. and Liashchynskiy, P., 2019. Grid search, random search, genetic algorithm: a big comparison for nas. *arXiv preprint arXiv:1912.06059*.
- Limna, P., 2022. Artificial intelligence (ai) in the hospitality industry: A review article. *International Journal of Computing Sciences Research. Advance online publication*, 6 (6), 1–12.
- Lin, D., Li, X. and Wang, D., 2011. Atavistic strategy for genetic algorithm. *International Conference in Swarm Intelligence*, Springer, 497–505.
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017a. Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.

- Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017b. Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, 2980–2988.
- Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017c. Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, 2980–2988.
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L., 2014. Microsoft coco: Common objects in context. *European conference on computer vision*, Springer, 740–755.
- Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A. L. and Fei-Fei, L., 2019a. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 82–92.
- Liu, G., Li, Y., Chen, Y., Shang, R. and Jiao, L., 2022a. Pol-nas: A neural architecture search method with feature selection for polsar image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15, 9339–9354.
- Liu, H., Li, Y., Duan, Z. and Chen, C., 2020a. A review on multi-objective optimization framework in wind energy forecasting techniques and applications. *Energy Conversion and Management*, 224, 113324.
- Liu, H., Simonyan, K. and Yang, Y., 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.
- Liu, J., Abbass, H. A. and Tan, K. C., 2019b. Evolutionary computation. *Evolutionary Computation and Complex Networks*, Springer, 3–22.
- Liu, J., Yuan, G., Yang, C., Song, H. and Luo, L., 2023. An interpretable cnn for the segmentation of the left ventricle in cardiac mri by real-time visualization. *CMES-Computer Modeling in Engineering & Sciences*, 135 (2).
- Liu, J., Zhang, K., Hu, W. and Yang, Q., 2022b. Improve ranking correlation of super-net through training scheme from one-shot nas to few-shot nas. *arXiv preprint arXiv:2206.05896*.
- Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J. and Han, J., 2019c. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.
- Liu, S. and Deng, W., 2015. Very deep convolutional neural network based image classification using small training sample size. *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*, IEEE, 730–734.

- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C., 2016. Ssd: Single shot multibox detector. *European conference on computer vision*, Springer, 21–37.
- Liu, W., Li, J., Zhao, G., Sun, L., Wang, H., Li, W. and Sun, B., 2020b. Improvement of cifar-10 image classification based on modified resnet-34. *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, Springer, 619–631.
- Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G. G. and Tan, K. C., 2021. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*.
- Liu, Z., Macuda, T., Xue, Z., Forsyth, D. S. and Laganière, R., 2010. Concealed weapon detection: A data fusion perspective. *Journal of Aerospace Computing, Information, and Communication*, 7 (7), 196–209.
- Long, J., Shelhamer, E. and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.
- Lorena, A. C., Garcia, L. P. F., Lehmann, J., Souto, M. C. P. and Ho, T. K., 2019. How complex is your classification problem? *ACM Computing Surveys*, 52 (5), 1–34. URL <http://dx.doi.org/10.1145/3347711>.
- Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60 (2), 91–110.
- Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E. and Banzhaf, W., 2018. Nsga-net: a multi-objective genetic algorithm for neural architecture search. *arXiv preprint arXiv:1810.03522*.
- Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E. and Banzhaf, W., 2019. Nsga-net: neural architecture search using multi-objective genetic algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference*, 419–427.
- Luc, D. T., 2008. Pareto optimality. *Pareto optimality, game theory and equilibria*, 481–515.
- Ma, W., Tu, X., Luo, B. and Wang, G., 2021. Semantic clustering based deduction learning for image recognition and classification. *Pattern Recognition*, 108440.
- Mahajan, R. and Padha, D., 2018. Detection of concealed weapons using image processing techniques: A review. *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, IEEE, 375–378.

- Majeed, R., Hatem, H. and Mohammed, M., 2018. Automatic detection system to the sticky bomb. *Computer Science and Engineering*.
- Malach, E. and Shalev-Shwartz, S., 2019. Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32.
- Marcos, D., Volpi, M. and Tuia, D., 2016. Learning rotation invariant convolutional filters for texture classification. *2016 23rd International Conference on Pattern Recognition (ICPR)*, IEEE, 2012–2017.
- Marr, D., 2010. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. MIT Press.
- Martin, T. and Koch, A., 2006. Recent developments in x-ray imaging with micrometer spatial resolution. *Journal of synchrotron radiation*, 13 (2), 180–194.
- Masruroh, S. U., Hardy, M. H. A., Hulliyah, K., Adelina, R. and Putri, R. A., 2023. Implementation of the cnn-svm hybrid model for leukocoria detection. *2023 4th International Conference on Big Data Analytics and Practices (IBDAP)*, IEEE, 1–6.
- Mattiussi, C., Marbach, D., Dürr, P. and Floreano, D., 2008. The age of analog networks. *AI Magazine*, 29 (3), 63–63.
- McLachlan, G. J., 1999. Mahalanobis distance. *Resonance*, 4 (6), 20–26.
- McNally, W., Vats, K., Wong, A. and McPhee, J., 2020. Evopose2d: Pushing the boundaries of 2d human pose estimation using neuroevolution. *arXiv preprint arXiv:2011.08446*.
- Mehta, Y., White, C., Zela, A., Krishnakumar, A., Zabergja, G., Moradian, S., Safari, M., Yu, K. and Hutter, F., 2022. Nas-bench-suite: Nas evaluation is (now) surprisingly easy. *arXiv preprint arXiv:2201.13396*.
- Mellor, J., Turner, J., Storkey, A. and Crowley, E. J., 2021. Neural architecture search without training. *International Conference on Machine Learning*, PMLR, 7588–7598.
- Mery, D., 2015. Computer vision for x-ray testing. *Switzerland: Springer International Publishing.–2015*, 10, 978–3.
- Mery, D. and Arteta, C., 2017. Automatic defect recognition in x-ray testing using computer vision. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, 1026–1035.
- Mery, D., Riffo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H. and Carrasco, M., 2015. Gdxray: The database of x-ray images for nondestructive testing. *Journal of Nondestructive Evaluation*, 34 (4), 1–12.

- Miao, C., Xie, L., Wan, F., Su, C., Liu, H., Jiao, J. and Ye, Q., 2019. Sixray: A large-scale security inspection x-ray benchmark for prohibited item discovery in overlapping images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2119–2128.
- Michel, S., Koller, S. M., de Ruiter, J. C., Moerland, R., Hogervorst, M. and Schwaninger, A., 2007. Computer-based training increases efficiency in x-ray image interpretation by aviation security screeners. *2007 41st Annual IEEE International Carnahan Conference on Security Technology*, IEEE, 201–206.
- Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N. et al., 2019. Evolving deep neural networks. *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, Elsevier, 293–312.
- Minhas, M. S. and Zelek, J., 2019. Anomaly detection in images. *arXiv preprint arXiv:1905.13147*.
- Mitchell, T. M., 2006. *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning
- Mitchell, T. M. et al., 1997. Machine learning.
- Moore, G. E. et al., 1965. Cramming more components onto integrated circuits.
- Morris, T., Chien, T. and Goodman, E., 2018. Convolutional neural networks for automatic threat detection in security x-ray images. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 285–292.
- Mouton, A. and Breckon, T. P., 2015. A review of automated image understanding within 3d baggage computed tomography security screening. *Journal of X-Ray Science and Technology*, 23 (5), 531–555.
- Munappy, A., Bosch, J., Olsson, H. H., Arpteg, A. and Brinne, B., 2019. Data management challenges for deep learning. *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, 140–147.
- Murphy, E. E., 1989. A rising war on terrorists. *IEEE Spectrum*, 26 (11), 33–36.
- Murphy, K. P., 2012. *Machine learning: a probabilistic perspective*. MIT press.
- Myers, G. J., Sandler, C. and Badgett, T., 2011. *The art of software testing*. John Wiley & Sons.
- Nacci, P. L. and Mockensturm, L., 2001. Detecting concealed weapons: Technology research at the national institute of justice. *Corrections Today*, 63 (4), 70–74.

- Nagarajah, T. and Poravi, G., 2019. A review on automated machine learning (automl) systems. *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, IEEE, 1–6.
- Nalçakan, Y. and Ensari, T., 2018. Decision of neural networks hyperparameters with a population-based algorithm. *International Conference on Machine Learning, Optimization, and Data Science*, Springer, 276–281.
- Naseri, H. and Mehrdad, V., 2023. Novel cnn with investigation on accuracy by modifying stride, padding, kernel size and filter numbers. *Multimedia Tools and Applications*, 1–19.
- Nath, S. S., Mishra, G., Kar, J., Chakraborty, S. and Dey, N., 2014. A survey of image classification methods and techniques. *2014 International conference on control, instrumentation, communication and computational technologies (ICCICCT)*, IEEE, 554–557.
- Nayyar, A., Garg, S., Gupta, D. and Khanna, A., 2018. Evolutionary computation: theory and algorithms. *Advances in swarm intelligence for optimizing problems in computer science*, Chapman and Hall/CRC, 1–26.
- Neishi, M., Sakuma, J., Tohda, S., Ishiwatari, S., Yoshinaga, N. and Toyoda, M., 2017. A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, 99–109.
- Ngatchou, P., Zarei, A. and El-Sharkawi, A., 2005. Pareto multi objective optimization. *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, IEEE, 84–91.
- Nguyen, H. D., Cai, R., Zhao, H., Kot, A. C. and Wen, B., 2022. Towards more efficient security inspection via deep learning: A task-driven x-ray image cropping scheme. *Micromachines*, 13 (4), 565.
- Nikolova, N. K. and McCombe, J. J., 2015. On-body concealed weapon detection system. US Patent App. 14/751,796.
- Nowozin, S., 2014. Optimal decisions from probabilistic models: the intersection-over-union case. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 548–555.
- Ogbolumani, O. A. and Nwulu, N. I., 2021. Multi-objective optimisation of constrained food-energy-water-nexus systems for sustainable resource allocation. *Sustainable Energy Technologies and Assessments*, 44, 100967.

- Olmos, R., Tabik, S. and Herrera, F., 2018. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, 275, 66–72.
- Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. and Kittler, J., 2010. A review of instance selection methods. *Artificial Intelligence Review*, 34 (2), 133–143.
- Operiano, K. R. G., Iba, H. and Pora, W., 2020. Neuroevolution architecture backbone for x-ray object detection. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2296–2303.
- Oussous, A., Benjelloun, F.-Z., Lahcen, A. A. and Belfkih, S., 2018. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30 (4), 431–448.
- Pampara, G., Engelbrecht, A. P. and Franken, N., 2006. Binary differential evolution. *2006 IEEE International Conference on Evolutionary Computation*, IEEE, 1873–1879.
- Papadopoulou, D., Sakalis, A., Merousis, N. and Tsirliganis, N., 2007. Study of decorated archeological ceramics by micro x-ray fluorescence spectroscopy. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 580 (1), 743–746.
- Papert, S., 1966. The summer vision project. memo aim-100.
- Parande, M. and Soma, S., 2015. Concealed weapon detection in a human body by infrared imaging. *International Journal of Science and Research (IJSR)*, 4 (9), 182–188.
- Parkhi, O. M., Vedaldi, A., Zisserman, A. and Jawahar, C., 2012. Cats and dogs. *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 3498–3505.
- Pasupa, K. and Sunhem, W., 2016. A comparison between shallow and deep architecture classifiers on small dataset. *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, IEEE, 1–6.
- Paszke, A., Chaurasia, A., Kim, S. and Culurciello, E., 2016. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.
- Patrick, M. K., Adekoya, A. F., Mighty, A. A. and Edward, B. Y., 2022. Capsule networks—a survey. *Journal of King Saud University-computer and information sciences*, 34 (1), 1295–1310.
- Pereira, F., Mitchell, T. and Botvinick, M., 2009. Machine learning classifiers and fmri: a tutorial overview. *Neuroimage*, 45 (1), S199–S209.

- Perez, F., Avila, S. and Valle, E., 2019. Solo or ensemble? choosing a cnn architecture for melanoma classification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 0–0.
- Petrozziello, A. and Jordanov, I., 2019. Automated deep learning for threat detection in luggage from x-ray images. *International Symposium on Experimental Algorithms*, Springer, 505–512.
- Pham, H., Guan, M., Zoph, B., Le, Q. and Dean, J., 2018. Efficient neural architecture search via parameters sharing. *International conference on machine learning*, PMLR, 4095–4104.
- Pllana, S., Memeti, S. and Kolodziej, J., 2019. Customizing pareto simulated annealing for multi-objective optimization of control cabinet layout. *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, IEEE, 78–85.
- Pons, J., Nieto, O., Prockup, M., Schmidt, E., Ehmann, A. and Serra, X., 2017. End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*.
- Pons, J., Pascual, S., Cengarle, G. and Serrà, J., 2021. Upsampling artifacts in neural audio synthesis. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 3005–3009.
- Prentis, P. J., Wilson, J. R., Dormontt, E. E., Richardson, D. M. and Lowe, A. J., 2008. Adaptive evolution in invasive species. *Trends in plant science*, 13 (6), 288–294.
- Pretorius, A., Van Biljon, E., Kroon, S. and Kamper, H., 2018. Critical initialisation for deep signal propagation in noisy rectifier neural networks. *arXiv preprint arXiv:1811.00293*.
- Price, K. V., 2013. Differential evolution. *Handbook of optimization*, Springer, 187–214.
- Qian, K., Zhang, Y., Chang, S., Yang, X. and Hasegawa-Johnson, M., 2019. Autovc: Zero-shot voice style transfer with only autoencoder loss. *International Conference on Machine Learning*, PMLR, 5210–5219.
- Qin, X. and Wang, Z., 2019. Nasnet: A neuron attention stage-by-stage net for single image deraining. *arXiv preprint arXiv:1912.03151*.
- Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Radiuk, P. and Kutucu, H., 2020. Heuristic architecture search using network morphism for chest x-ray classification. *IntelliTISIS*, 107–121.

- Radiuk, P. M. et al., 2017. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, 20 (1), 20–24.
- Rashid, T., 2016. *Make your own neural network*. CreateSpace Independent Publishing Platform.
- Real, E., Aggarwal, A., Huang, Y. and Le, Q. V., 2018. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*.
- Real, E., Aggarwal, A., Huang, Y. and Le, Q. V., 2019. Regularized evolution for image classifier architecture search. *Proceedings of the aaai conference on artificial intelligence*, volume 33, 4780–4789.
- Real, E., Liang, C., So, D. and Le, Q., 2020. Automl-zero: Evolving machine learning algorithms from scratch. *International conference on machine learning*, PMLR, 8007–8019.
- Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V. and Kurakin, A., 2017. Large-scale evolution of image classifiers. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2902–2911.
- Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.
- Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 91–99.
- Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. and Savarese, S., 2019. Generalized intersection over union: A metric and a loss for bounding box regression. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 658–666.
- Riesenhuber, M. and Poggio, T., 2000. Models of object recognition. *Nature neuroscience*, 3 (11), 1199–1204.
- Rodellar, J., Barrera, K., Alférez, S., Boldú, L., Laguna, J., Molina, A. and Merino, A., 2022. A deep learning approach for the morphological recognition of reactive lymphocytes in patients with covid-19 infection. *Bioengineering*, 9 (5), 229.
- Rodriguez, P. L., Spirling, A. and Stewart, B. M., 2023. Embedding regression: Models for context-specific description and inference. *American Political Science Review*, 1–20.

- Rolnick, D., Veit, A., Belongie, S. and Shavit, N., 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.
- Ronald, E. and Schoenauer, M., 1994. Genetic lander: An experiment in accurate neuro-genetic control. *International Conference on Parallel Problem Solving from Nature*, Springer, 452–461.
- Ronneberger, O., Fischer, P. and Brox, T., 2015a. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, Springer, 234–241.
- Ronneberger, O., Fischer, P. and Brox, T., 2015b. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, Springer, 234–241.
- Röntgen, W., 1895. The x-rays: Upon a new kind of rays. *Annual Report of the Board of Regents of the Smithsonian Institution*.
- Rose, L. T. and Fischer, K. W., 2011. Garbage in, garbage out: Having useful data is everything. *Measurement: Interdisciplinary Research & Perspective*, 9 (4), 222–226.
- Rosenblatt, F., 1957. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.
- Rostami, S., 2014. *Preference focussed many-objective evolutionary computation*. Ph.D. thesis, Manchester Metropolitan University.
- Rostami, S., Kleszcz, A., Dimanov, D. and Katos, V., 2020. A machine learning approach to dataset imputation for software vulnerabilities. *International Conference on Multimedia Communications, Services and Security*, Springer, 25–36.
- Rostami, S. and Neri, F., 2016. Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. *Integrated Computer-Aided Engineering*, 23 (4), 313–329.
- Rostami, S., O'Reilly, D., Shenfield, A. and Bowring, N., 2015. A novel preference articulation operator for the evolutionary multi-objective optimisation of classifiers in concealed weapons detection. *Information Sciences*, 295, 494–520.
- Rostami, S. and Shenfield, A., 2012. Cma-paes: Pareto archived evolution strategy using covariance matrix adaptation for multi-objective optimisation. *2012 12th UK Workshop on Computational Intelligence (UKCI)*, IEEE, 1–8.
- Rothlauf, F., 2006. Representations for genetic and evolutionary algorithms. *Representations for Genetic and Evolutionary Algorithms*, Springer, 9–32.

- Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al., 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115 (3), 211–252.
- Russo, D. and Zou, J., 2019. How much does your data exploration overfit? controlling bias via information usage. *IEEE Transactions on Information Theory*, 66 (1), 302–323.
- Sabour, S., Frosst, N. and Hinton, G. E., 2017. Dynamic routing between capsules. *Advances in neural information processing systems*, 3856–3866.
- Safe, M., Carballido, J., Ponzoni, I. and Brignole, N., 2004. On stopping criteria for genetic algorithms. *Brazilian Symposium on Artificial Intelligence*, Springer, 405–413.
- Sagar, A. and Soundrapandiyam, R., 2020. Semantic segmentation with multi scale spatial attention for self driving cars. *arXiv preprint arXiv:2007.12685*.
- Sanders, H. and Saxe, J., 2017. Garbage in, garbage out: how purportedly great ml models can be screwed up by bad data. *Proceedings of Blackhat*, 2017.
- Sandler, T. and Smith, V. K., 1982. Intertemporal and intergenerational pareto efficiency: A reconsideration of recent extensions. *Journal of Environmental Economics and Management*, 9 (4), 361–365.
- Sanida, M. V., Sanida, T., Sideris, A. and Dasygenis, M., 2023. An efficient hybrid cnn classification model for tomato crop disease. *Technologies*, 11 (1), 10.
- Sarkar, A., Vijaykeerthy, D., Sarkar, A. and Balasubramanian, V. N., 2022. A framework for learning ante-hoc explainable models via concepts. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10286–10295.
- Scherer, D., Müller, A. and Behnke, S., 2010. Evaluation of pooling operations in convolutional architectures for object recognition. *International conference on artificial neural networks*, Springer, 92–101.
- Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural networks*, 61, 85–117.
- Schraudolph, N. N. and Belew, R. K., 1992. Dynamic parameter encoding for genetic algorithms. *Machine learning*, 9 (1), 9–21.
- Sener, O. and Savarese, S., 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

- Seneviratne, S., Senanayake, D., Rasnayaka, S., Vidanaarachchi, R. and Thompson, J., 2022. Dalle-urban: Capturing the urban design expertise of large text to image transformers. *2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, 1–9.
- Seo, Y. and Shin, K.-s., 2019. Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, 116, 328–339.
- Sereno, D. M. D., 2018. *Automatic Evolution of Deep AutoEncoders*. Ph.D. thesis, Universidade de Coimbra.
- Settles, B., 2009. Active learning literature survey.
- Shaban, A., Bansal, S., Liu, Z., Essa, I. and Boots, B., 2017. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*.
- Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. and De Freitas, N., 2015. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1), 148–175.
- Shaw, A., Hunter, D., Landola, F. and Sidhu, S., 2019. Squeezenas: Fast neural architecture search for faster semantic segmentation. *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 0–0.
- Sheen, D. M., McMakin, D. L. and Hall, T. E., 2001. Three-dimensional millimeter-wave imaging for concealed weapon detection. *IEEE Transactions on microwave theory and techniques*, 49 (9), 1581–1592.
- Shen, W., Li, G., Wei, X., Fu, Q., Zhang, Y., Qu, T., Chen, C. and Wang, R., 2022. Assessment of dairy cow feed intake based on bp neural network with polynomial decay learning rate. *Information Processing in Agriculture*, 9 (2), 266–275.
- Shen, X., Wang, Y., Lin, M., Huang, Y., Tang, H., Sun, X. and Wang, Y., 2023. Deepmad: Mathematical architecture design for deep convolutional neural network. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6163–6173.
- Shenfield, A. and Rostami, S., 2017. Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. *2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, IEEE, 1–8.
- Shin, R., Packer, C. and Song, D., 2018. Differentiable neural network architecture search.

- Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S. and Jagersand, M., 2018. Rt-seg: Real-time semantic segmentation comparative study. *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 1603–1607.
- Siddiqi, A. A. and Lucas, S. M., 1998. A comparison of matrix rewriting versus direct encoding for evolving neural networks. *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, IEEE, 392–397.
- Sidky, E. Y., Duchin, Y., Pan, X. and Ullberg, C., 2011. A constrained, total-variation minimization algorithm for low-intensity x-ray ct. *Medical physics*, 38 (S1), S117–S125.
- Siebel, N. T., Botel, J. and Sommer, G., 2009. Efficient neural network pruning during neuro-evolution. *2009 International Joint Conference on Neural Networks*, IEEE, 2920–2927.
- Siems, J. N., Zimmer, L., Zela, A., Lukasik, J., Keuper, M. and Hutter, F., 2020. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search.
- Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Singaravel, S., Suykens, J., Janssen, H. and Geyer, P., 2020. Explainable deep convolutional learning for intuitive model development by non-machine learning domain experts. *Design Science*, 6.
- Singh, K. K. and Lee, Y. J., 2017a. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. *2017 IEEE international conference on computer vision (ICCV)*, IEEE, 3544–3553.
- Singh, K. K. and Lee, Y. J., 2017b. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. *2017 IEEE international conference on computer vision (ICCV)*, IEEE, 3544–3553.
- Smith, L. N., 2017. Cyclical learning rates for training neural networks. *2017 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 464–472.
- Smith, L. N., 2018. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.
- Smith, S. L., Kindermans, P.-J., Ying, C. and Le, Q. V., 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

- Solorio-Fernández, S., Carrasco-Ochoa, J. A. and Martínez-Trinidad, J. F., 2020. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53 (2), 907–948.
- Sriram, B., Ghaffari, A., Rajagopal, K., Jafari, S. and Tlelo-Cuautle, E., 2023. A chaotic map with trigonometric functions: Dynamical analysis and its application in image encryption based on sparse representation and convolutional filters. *Optik*, 273, 170379.
- Stanley, K., 2003. Miikkulainen. a taxonomy for artificial embryogeny. *Artificial Life*, 9 (2), 93–130.
- Stanley, K. O., Clune, J., Lehman, J. and Miikkulainen, R., 2019a. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1 (1), 24–35.
- Stanley, K. O., Clune, J., Lehman, J. and Miikkulainen, R., 2019b. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1 (1), 24–35.
- Stanley, K. O., D'Ambrosio, D. B. and Gauci, J., 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15 (2), 185–212.
- Stanley, K. O. and Miikkulainen, R., 2002a. Efficient evolution of neural network topologies. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, IEEE, volume 2, 1757–1762.
- Stanley, K. O. and Miikkulainen, R., 2002b. Efficient reinforcement learning through evolving neural network topologies. *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, Morgan Kaufmann Publishers Inc., 569–577.
- Sun, M., Song, Z., Jiang, X., Pan, J. and Pang, Y., 2017. Learning pooling for convolutional neural network. *Neurocomputing*, 224, 96–104.
- Sun, S.-H., Fan, S.-P. and Wang, Y.-C. F., 2014. Exploiting image structural similarity for single image rain removal. *2014 IEEE International Conference on Image Processing (ICIP)*, IEEE, 4482–4486.
- Sun, Y., Xue, B., Zhang, M. and Yen, G. G., 2019. Evolving deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 24 (2), 394–407.
- Surya, L., 2015. An exploratory study of ai and big data, and it's future in the united states. *International Journal of Creative Research Thoughts (IJCRT)*, ISSN, 2320–2882.
- Sutskever, I., Martens, J., Dahl, G. and Hinton, G., 2013. On the importance of initialization and momentum in deep learning. *International conference on machine learning*, PMLR, 1139–1147.

- Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. A., 2017a. Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-First AAAI Conference on Artificial Intelligence*, 4278–4284.
- Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. A., 2017b. Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-first AAAI conference on artificial intelligence*.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015a. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015b. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015c. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016a. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016b. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.
- Szeliski, R., 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.
- Szeliski, R., 2011. *Computer vision, texts in computer science*.
- Szumner, M. and Picard, R. W., 1998. Indoor-outdoor image classification. *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, IEEE, 42–51.
- Taghanaki, S. A., Abhishek, K., Cohen, J. P., Cohen-Adad, J. and Hamarneh, G., 2021. Deep semantic segmentation of natural and medical images: a review. *Artificial Intelligence Review*, 54 (1), 137–178.
- Talwalkar, A., Kumar, S. and Rowley, H., 2008. Large-scale manifold learning. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1–8.

- Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A. and Le, Q. V., 2019. Mnasnet: Platform-aware neural architecture search for mobile. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2820–2828.
- Tan, M., Hartley, M., Bister, M. and Deklerck, R., 2009. Automated feature selection in neuroevolution. *Evolutionary Intelligence*, 1, 271–292.
- Tan, M. and Le, Q., 2019a. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, PMLR, 6105–6114.
- Tan, M. and Le, Q. V., 2019b. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.
- Tan, M., Pang, R. and Le, Q. V., 2020. Efficientdet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10781–10790.
- Tanaka, H., Kunin, D., Yamins, D. L. and Ganguli, S., 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*.
- Tang, R. and Lin, J., 2018. Deep residual learning for small-footprint keyword spotting. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 5484–5488.
- Tapia, M. G. C. and Coello, C. A. C., 2007. Applications of multi-objective evolutionary algorithms in economics and finance: A survey. *2007 IEEE Congress on Evolutionary Computation*, IEEE, 532–539.
- Taqi, A. M., Awad, A., Al-Azzo, F. and Milanova, M., 2018. The impact of multi-optimizers and data augmentation on tensorflow convolutional neural network performance. *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, IEEE, 140–145.
- Targ, S., Almeida, D. and Lyman, K., 2016. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*.
- Teh, E. W. and Taylor, G. W., 2020. Learning with less data via weakly labeled patch classification in digital pathology. *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, IEEE, 471–475.
- Theis, L., Shi, W., Cunningham, A. and Huszár, F., 2017. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*.
- Tiwari, R. K. and Verma, G. K., 2015. A computer vision based framework for visual gun detection using harris interest point detector. *Procedia Computer Science*, 54, 703–712.

- Tomoiagă, B., Chindriș, M., Sumper, A., Sudria-Andreu, A. and Villafafila-Robles, R., 2013. Pareto optimal reconfiguration of power distribution systems using a genetic algorithm based on nsga-ii. *Energies*, 6 (3), 1439–1455.
- Tompson, J., Goroshin, R., Jain, A., LeCun, Y. and Bregler, C., 2015. Efficient object localization using convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 648–656.
- Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y. and Gordon, G. J., 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.
- Torrey, L. and Shavlik, J., 2010. Transfer learning. *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, 242–264.
- Triguero, I., Galar, M., Vluymans, S., Cornelis, C., Bustince, H., Herrera, F. and Saeys, Y., 2015. Evolutionary undersampling for imbalanced big data classification. *2015 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 715–722.
- Tsamardinos, I., Charonyktakis, P., Papoutsoglou, G., Borboudakis, G., Lakiotaki, K., Zenklusen, J. C., Juhl, H., Chatzaki, E. and Lagani, V., 2022. Just add data: automated predictive modeling for knowledge discovery and feature selection. *NPJ precision oncology*, 6 (1), 38.
- Tsang, I. W., Kwok, J. T., Cheung, P.-M. and Cristianini, N., 2005. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6 (4).
- Tschiatschek, S., Iyer, R. K., Wei, H. and Bilmes, J. A., 2014. Learning mixtures of sub-modular functions for image collection summarization. *Advances in neural information processing systems*, 27.
- Turner, J., Crowley, E. J. and O’Boyle, M. F., 2021. Neural architecture search as program transformation exploration. *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 915–927.
- Valvano, G., Leo, A., Della Latta, D., Martini, N., Santini, G., Chiappino, D. and Ricciardi, E., 2018. Unsupervised data selection for supervised learning. *arXiv preprint arXiv:1810.12142*.
- Van Laarhoven, P. J. and Aarts, E. H., 1987. Simulated annealing. *Simulated annealing: Theory and applications*, Springer, 7–15.
- Varano, C., 2017. Disentangling variational autoencoders for image classification. *cs231n.stanford.edu*.

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Venkataramanan, S., Kijak, E., Amsaleg, L. and Avrithis, Y., 2022. Alignmixup: Improving representations by interpolating aligned features. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19174–19183.
- Vercruyssen, V., Meert, W. and Davis, J., 2017. Transfer learning for time series anomaly detection. *Proceedings of the Workshop and Tutorial on Interactive Adaptive Learning@ ECMLPKDD 2017*, CEUR Workshop Proceedings, volume 1924, 27–37.
- Viola, P. and Jones, M. J., 2004. Robust real-time face detection. *International journal of computer vision*, 57 (2), 137–154.
- Wan, A., Dunlap, L., Ho, D., Yin, J., Lee, S., Jin, H., Petryk, S., Bargal, S. A. and Gonzalez, J. E., 2020. Nbd: Neural-backed decision trees. *arXiv preprint arXiv:2004.00221*.
- Wang, B., Zhang, L., Wen, L., Liu, X. and Wu, Y., 2021. Towards real-world prohibited item detection: A large-scale x-ray benchmark. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5412–5421.
- Wang, T., Zhu, J.-Y., Torralba, A. and Efros, A. A., 2018. Dataset distillation. *arXiv preprint arXiv:1811.10959*.
- Wang, W., Huang, Y., Wang, Y. and Wang, L., 2014. Generalized autoencoder: A neural network framework for dimensionality reduction. *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 490–497.
- Wang, W., Lee, C. M., Liu, J., Colakoglu, T. and Peng, W., 2023. An empirical study of cyclical learning rate on neural machine translation. *Natural Language Engineering*, 29 (2), 316–336.
- Watanabe, S., 2013. A widely applicable bayesian information criterion. *Journal of Machine Learning Research*, 14 (Mar), 867–897.
- Wei, K., Iyer, R. and Bilmes, J., 2015. Submodularity in data subset selection and active learning. *International conference on machine learning*, PMLR, 1954–1963.
- Wei, Q., Ren, Y., Hou, R., Shi, B., Lo, J. Y. and Carin, L., 2018. Anomaly detection for medical images based on a one-class classification. *Medical Imaging 2018: Computer-Aided Diagnosis*, SPIE, volume 10575, 375–380.
- Weiss, S. M. and Kulikowski, C. A., 1991. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*, volume 123. Morgan Kaufmann San Mateo, CA.

- Weng, Y., Zhou, T., Li, Y. and Qiu, X., 2019. Nas-unet: Neural architecture search for medical image segmentation. *IEEE access*, 7, 44247–44257.
- Wetter, O. E., 2013. Imaging in airport security: Past, present, future, and the link to forensic and clinical radiology. *Journal of Forensic Radiology and Imaging*, 1 (4), 152–160.
- White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D. and Hutter, F., 2023. Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*.
- Wilson, D. L., 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3), 408–421.
- Woolley, B. G. and Stanley, K. O., 2014. A novel human-computer collaboration: combining novelty search with interactive evolution. *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, ACM, 233–240.
- Wu, D., Ying, Y., Zhou, M., Pan, J. and Cui, D., 2023. Improved resnet-50 deep learning algorithm for identifying chicken gender. *Computers and Electronics in Agriculture*, 205, 107622.
- Wu, H., Zhang, J., Huang, K., Liang, K. and Yu, Y., 2019a. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv preprint arXiv:1903.11816*.
- Wu, J., 2017. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5–23.
- Wu, M.-T., Lin, H.-I. and Tsai, C.-W., 2021. A training-free genetic neural architecture search. *Proceedings of the 2021 ACM International Conference on Intelligent Computing and its Emerging Applications*, 65–70.
- Wu, R., Yan, S., Shan, Y., Dang, Q. and Sun, G., 2015. Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*.
- Wu, Z., Shen, C. and Van Den Hengel, A., 2019b. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90, 119–133.
- Xia, R., Li, G., Huang, Z., Wen, L. and Pang, Y., 2021. Classify and localize threat items in x-ray imagery with multiple attention mechanism and high-resolution and high-semantic features. *IEEE Transactions on Instrumentation and Measurement*, 70, 1–10.
- Xia, X., Xu, C. and Nan, B., 2017. Inception-v3 for flower classification. *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, IEEE, 783–787.

- Xiao, H., Rasul, K. and Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.
- Xie, L. and Yuille, A., 2017. Genetic cnn. *Proceedings of the IEEE International Conference on Computer Vision*, 1379–1388.
- Xu, H., Yao, L., Zhang, W., Liang, X. and Li, Z., 2019. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. *Proceedings of the IEEE/CVF international conference on computer vision*, 6649–6658.
- Xu, Y., Cheng, L., Cai, X., Ma, X., Chen, W., Zhang, L. and Wang, Y., 2023. Efficient supernet training using path parallelism. *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, IEEE, 1249–1261.
- Xue, C., Yan, J., Yan, R., Chu, S. M., Hu, Y. and Lin, Y., 2019. Transferable automl by model sharing over grouped datasets. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9002–9011.
- Xue, Z. and Blum, R. S., 2003. Concealed weapon detection using color image fusion. *Proceedings of the 6th international conference on information fusion*, volume 1, 622–627.
- Yan, S., Zheng, Y., Ao, W., Zeng, X. and Zhang, M., 2020. Does unsupervised architecture representation learning help neural architecture search? *arXiv preprint arXiv:2006.06936*.
- Yang, S., 2007. Genetic algorithms with elitism-based immigrants for changing optimization problems. *Workshops on Applications of Evolutionary Computation*, Springer, 627–636.
- Yang, X., De Carlo, F., Phatak, C. and Gürsoy, D., 2017a. A convolutional neural network approach to calibrating the rotation axis for x-ray computed tomography. *Journal of Synchrotron Radiation*, 24 (2), 469–475.
- Yang, X., Yu, L., Li, S., Wang, X., Wang, N., Qin, J., Ni, D. and Heng, P.-A., 2017b. Towards automatic semantic segmentation in volumetric ultrasound. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 711–719.
- Yang, Z., Chen, Y. and Zhang, S., 2021. Recognition of design fixation via body language using computer vision. *Mathematical Problems in Engineering*, 2021.
- Yao, Q., Wang, M., Chen, Y., Dai, W., Yi-Qi, H., Yu-Feng, L., Wei-Wei, T., Qiang, Y. and Yang, Y., 2018. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*.

- Ye, Y., Wu, Q., Huang, J. Z., Ng, M. K. and Li, X., 2013. Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, 46 (3), 769–787.
- Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K. and Hutter, F., 2019. Nas-bench-101: Towards reproducible neural architecture search. *International Conference on Machine Learning*, PMLR, 7105–7114.
- Yoo, D., Park, S., Lee, J.-Y. and So Kweon, I., 2015. Multi-scale pyramid pooling for deep convolutional representation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 71–80.
- Yu, K., Ranftl, R. and Salzmann, M., 2021. An analysis of super-net heuristics in weight-sharing nas. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (11), 8110–8124.
- Yuan, Y., Xu, H. and Wang, B., 2014. An improved nsga-iii procedure for evolutionary many-objective optimization. *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, 661–668.
- Zagoruyko, S. and Komodakis, N., 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.
- Zapotecas-Martínez, S., Coello, C. A. C., Aguirre, H. E. and Tanaka, K., 2018. A review of features and limitations of existing scalable multiobjective test suites. *IEEE Transactions on Evolutionary Computation*, 23 (1), 130–142.
- Zeiler, M. D. and Fergus, R., 2014. Visualizing and understanding convolutional networks. *European conference on computer vision*, Springer, 818–833.
- Zhang, H., Liu, H. and Gu, Y., 2023a. Msfnets: A multi-level semantic feature fusion network for instance segmentation of x-ray security images. *2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, IEEE, volume 3, 1459–1464.
- Zhang, J., 2022. A novel intelligent radar detection network. *Journal of Physics: Conference Series*, IOP Publishing, volume 2181, 012056.
- Zhang, Q., Cui, Z., Niu, X., Geng, S. and Qiao, Y., 2017a. Image segmentation with pyramid dilated convolution based on resnet and u-net. *International conference on neural information processing*, Springer, 364–372.
- Zhang, Q. and Li, H., 2007. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11 (6), 712–731.

- Zhang, Q., Xiao, S., Huang, R., Zhang, R. and Zeng, P., 2023b. Research on visual recognition method of distribution network grounding ring. *International Conference on Computer Network Security and Software Engineering (CNSSE 2023)*, SPIE, volume 12714, 427–432.
- Zhang, S., Benenson, R. and Schiele, B., 2017b. Citypersons: A diverse dataset for pedestrian detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3221.
- Zhang, Y., 2018. A better autoencoder for image: Convolutional autoencoder. *ICONIP17-DCEC*. Available online: http://users.cecs.anu.edu.au/Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58.pdf (accessed on 23 March 2017).
- Zhang, Y. and Wu, L., 2012. Classification of fruits using computer vision and a multiclass support vector machine. *sensors*, 12 (9), 12489–12505.
- Zhou, D., Zhou, X., Zhang, W., Loy, C. C., Yi, S., Zhang, X. and Ouyang, W., 2020. Econas: Finding proxies for economical neural architecture search. *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 11396–11404.
- Zhu, M., Xia, J., Jin, X., Yan, M., Cai, G., Yan, J. and Ning, G., 2018. Class weights random forest algorithm for processing class imbalanced medical data. *IEEE Access*, 6, 4641–4652.
- Zitzler, E., Brockhoff, D. and Thiele, L., 2007. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 862–876.
- Zoph, B. and Le, Q. V., 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.
- Zoph, B., Vasudevan, V., Shlens, J. and Le, Q. V., 2018. Learning transferable architectures for scalable image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.