Multimedia Tools and Applications (2023) 82:1519–1536
https://doi.org/10.1007/s11042-022-13269-1

TRACK 1: GENERAL MULTIMEDIA TOPICS

Uncertainty estimation based adversarial attack in multi-class classification

Ismail Alarab[1] · Simant Prakoonwit[1]

Received: 11 March 2022 / Revised: 11 May 2022 / Accepted: 23 May 2022 /
Published online: 3 June 2022

Abstract
Model uncertainty has gained popularity in machine learning due to the overconfident predictions derived from standard neural networks which are not trustworthy. Recently, Monte-Carlo based adversarial attack (MC-AA) has been proposed as a simple uncertainty estimation method which is powerful in capturing data points that lie in the overlapping distribution of the decision boundary. MC-AA produces uncertainties by performing back-and-forth perturbations of a given data point towards the decision boundary using the idea of adversarial attacks. Despite its efficacy against other uncertainty estimation methods, this method has been only examined on binary classification problems. Thus, we present and examine MC-AA with multi-class classification tasks. We point out the limitation of this method with multiple classes which we tackle by converting multiclass problem into 'one-versus-all' classification. We compare MC-AA against other recent model uncertainty methods on Cora – a graph structured dataset – and MNIST – an image dataset. Thus, the conducted experiments are performed using a variety of deep learning algorithms to perform the classification. Consequently, we discuss the best results of model uncertainty with Cora data using LEConv model of AUC-score 0.889 and MNIST data using CNN of AUC-score 0.98 against other uncertainty estimation methods.

Keywords Uncertainty estimation · Adversarial attack · Deep neural network

1 Introduction

Machine learning applications have been exhaustively attracting the interests of many in the globe with various applications such as healthcare [6, 10, 17], blockchain [3, 41], cyber-

✉ Ismail Alarab
ialarab@bournemouth.ac.uk

Simant Prakoonwit
sprakoonwit@bournemouth.ac.uk

[1] Bournemouth University, Bournemouth, UK

security [9, 16], and self-driving cars [25, 32]. On the other hand, machine learning models accompanied with softmax outputs often produces overconfident predictions which lead to poor decision-making regardless the models' performance. In Fig. 1, an image of digit "2″ in the MNIST dataset is predicted as "7″ with high confidence. Indeed, it is obvious to interpret this confusion by the machine learning model wherein the machine learning confidently misclassifies the given example. There are more serious scenarios that provide erroneous and confident predictions leading to more cataclysmic decisions such the case in self-driving cars [36]. In May 2016, a fatal accident is caused by the autopilot feature of a Tesla Model S, which failed to discriminate the white tractor-trailer against a bright sky. Furthermore, other applications are also subjected to misleading predictions such as in misclassifying legitimate transactions as illegal in anti-money laundering [4]. These faults caused by the learning model should be detected, analysed, and avoided. Consequently, model uncertainty is unavoidably needed besides the model's predictions to provide a reliable decision-making. Model Uncertainty in machine learning has interestingly become an emergent topic by the resurgence of Bayesian approximation in [12]. Regarding classification tasks, there are two main types of uncertainty known as epistemic and aleatoric uncertainty. Epistemic occurs when a new data point lacks to in formation of training instances, whereas aleatoric is observed when data points fall in common regions of different class distribution resulting in wrong predictions. The former can be reduced by more data, while the latter is unavoidable unless more informative features are added to obtain a better classification rule. Gal et al. [12] have tackled the computational complexity of Bayesian neural networks by proposing Monte-Carlo dropout – a Bayesian approximation method – which efficiently produce uncertainty estimates in machine learning models. A comprehensive review covering the recent advances in uncertainty quantification methods is carried out by Abdar et al. [1]. Concisely, uncertainty quantification methods have appeared to detect data points that the model is not trained on such in deterministic uncertainty quantification (DUQ) [37], or to capture data points that fall near decision boundary such in Deep Ensembles [20] and Monte-Carlo dropout (MC-dropout) [13]. DUQ reflects the out-of-distribution tested points which appear far from the trained data. Deep Ensembles and MC-dropout are commonly based on ensemble of models to produce uncertainty. However, the former method is a combination of model with different initialised parameters. Whereas the latter one is an ensemble of models with shared parameters by the means of the dropout. Despite their efficiency in modelling uncertainty, these methods have revealed a significant drawback [2, 5]. It is identified by the failure of these methods in detecting the points that fall in the overlapping regions. Henceforth, the data points falling in the overlapping regions of multiple class distributions cannot be influenced by the variability of the decision boundary referring to [2]. The latter study has introduced Monte-Carlo adversarial attack (MC-AA), an uncertainty method that provides perturbations on the given data point in the direction of the decision boundary. The perturbed inputs are computed using the adversarial attack method where multiple perturbed input samples are linearly generated. Consequently, these samples produce uncertainty in the same experimental procedure to MC-dropout. MC-AA has shown its capability in capturing model uncertainty on data points by forcing them to travel between two given class distributions. Despite its promising performance, MC-AA has been only studied with binary classification problems. Motivated by previous work in [2], we propose a general way of MC-AA method that can be applied to multi-class classifications. MC-AA in multiclass classification is a challenging problem because there exist multiple decision boundaries. In other words, it is not clear in which directions the multiple back-and-forth perturbations on a given data point should be performed.

In this study, we propose a generalised MC-AA that performs multiple perturbations in back-and-forth fashion towards the decision boundary that is associated with the predicted class on a given data point. Subsequently, we conduct our experiments using a variety of deep learning models on Cora and MNIST datasets as a type of multi-classification problems. Then, we evaluate and compare the performance of the model uncertainty using MC-AA against recent uncertainty estimation methods. On the other hand, the presented MC-AA has revealed limitations in some cases, which we discuss in the experiments. We effectively solve these limitations by converting multi-class classification to one-versus-all binary classification. Admittedly, we show the competence of the presented MC-AA in capturing uncertainty against other uncertainty methods on the given datasets. This paper is structurally divided as follows: Section 2 involves the overview of the related work. Section 3 provides the background of the previously proposed MC-AA on binary classification problems. Section 4 demonstrates our generalised MC-AA with multiclass classification tasks. Section 5 provides the conducted experiments and results. A discussion and a conclusion are given in Section 6 and 7, respectively.

2 Overview of related works

Primarily, neural networks with distributions over the weights have emerged as Bayesian neural networks (BNNs) that have been studied by Neal et al. [28, 29] and by Mackay et al. [24]. Subsequently, BNN models have witnessed resurgence in the recent years referring to [7, 15, 18]. Admittedly, BNNs have revealed significant success in modelling uncertainty of neural networks. However, this approach is subjected to prohibitive computational cost referring [12]. Consequently, Gal et al. [13] have proposed MC-dropout method as an approximation of Bayesian approach which uses dropout as a variational inference. MC-dropout is a simple and efficient method that uses dropout [35] after each hidden layer to produce uncertainty estimates in neural networks [19]. To produce uncertainty, this is performed using dropout during testing phase [13]; thus, a data point is subjected to multiple stochastic versions of perturbed decision boundary which reflects the uncertainty about its

Fig. 1 Example of digit two image in MNIST dataset

predictions. Subsequently, an adaptive version of MC-dropout has appeared in [14] where the dropout parameter is optimised with respect to a given objective function. *Deep Ensemble* method is another Bayesian approximation which utilises an ensemble of multiple neural network models with different initialisations [20]. The study in [30] has shown that Deep Ensemble method has a superior success over BNNs. However, this method has shown poor performance on a simple 2D synthetic data [37]. The latter study has introduced deterministic uncertainty quantification (DUQ) model which reliably captures the out-of-distribution data – i.e., data points distant from the trained data. DUQ is a deep model that learns feature representations in which the distance between these features and centroids derived from the training data are assessed using a kernel function. This model that uses radial basis function (RBF) kernel is known as RBF network [22]. Other uncertainty estimation methods also exist such as using an approximated variational inference by Gaussian processes in [38], *DropConnect* as another version of MC-dropout [27] and uncertainty estimation based on evidential deep learning [40]. MC-dropout and Deep Ensemble methods seek to perturb the decision boundary between different class distributions where they have revealed promising results in capturing model uncertainties. However, these methods have failed to capture data points that fall in the overlapping region of class distributions. This issue has been tackled in [2] by proposing MC-AA. MC-AA is an uncertainty estimation method that uses adversarial attack idea to perform back-and-forth perturbations of a given data point toward the decision boundary. Primarily, adversarial attacks have been extensively discovered in various aspects of machine learning such as in [23] to improve classification, in [34, 39] to act against adversarial examples. However, MC-AA is used to spot out data points lying near decision boundary of neural network models, wherein noisy points can be detected with high uncertainty. MC-AA has revealed significant outperformance over other methods in producing reliable predictive uncertainties in binary classification problems [2]. Thus, we conduct experiments on multiclass classification datasets using various deep learning model to capture model uncertainty using MC-AA. Furthermore, we compare the model performance against MC-dropout, DUQ and Deep Ensemble methods.

3 Methods for quantifying uncertainty

In this section, we present the methods used in our experiments to quantify uncertainty of deep learning models.

3.1 Monte-Carlo based adversarial attack: MC-AA

Adversarial attacks are crafted inputs to fool the neural network decision [8]. Obtaining adversarial example in white box attacks using Fast Gradient Sign Method (FGSM) can be expressed as:

$$x_{Adv} = x + \epsilon.\text{sign}(\nabla_x J(x, y)), \tag{1}$$

where x_{Adv} is the crafted input known as an adversarial example, ϵ is a small scale between 0 and 1, ∇_x is the gradient with respect to the initial input x, and y is the desired class label. Moreover, *sign* is the sign function that produces 1 for positive values and -1 for negative ones.

The study in [2] has proposed MC-AA, an uncertainty estimation method based on the idea adversarial attacks, which produces uncertainty estimates besides the predictions of a neural network. MC-AA uses FGSM method to perturb the inputs during the testing phase with multiple values of ϵ that belong to a small symmetric interval in a neighbourhood of zero. However, FGSM requires the desired class label to shift the data input in the opposite direction of the assigned class. In [2], MC-AA assigns an arbitrary class label (i.e., 0 or 1) to FGSM to its given inputs, wherein multiple perturbed versions of each input are produced derived from multiple values of ϵ.

Algorithm 1 Generalised MC-AA

Require:

- $\mathcal{M}$: is a neural network that maps the feature vectors to multiclass predictions.
- $J(.,.)$: is a loss function (e.g., binary cross entropy) that requires as input the predicted class derived from a given data point and a desired class label.
- **Input:**
  - **I:** is a discrete interval that consists of a set of ϵ values such that $I = \{-\epsilon_{max}, \ldots, 0, 0+\beta, \ldots, \epsilon_{max}\}$, which is evenly spaced by β and centered around zero. ϵ_{max} is small scalar to be tuned.
  - x: is an input feature vector.
  - $c = argmax(M(x))$ is the predicted class.
- **Output:**
  - $\hat{y}$: is a set of outputs for a single data point that is subjected to multiple perturbations. $\hat{y} = \{\hat{y}_{\epsilon_1}, \ldots, \hat{y}_{\epsilon_n}\}$, such that $\epsilon_i \in I$.

Function

Compute the gradients of the loss function with respect to x as:
grad = $\nabla_x J(\mathcal{M}(x), c)$
for all $\epsilon_i \in I$ **do**

$x_{\epsilon_i} \leftarrow x + \epsilon_i . sign(grad)$;
$\hat{y}_{\epsilon_i} \leftarrow \mathcal{M}(x_{\epsilon_i})$;
return $\hat{y}_{\epsilon_i}$

end for
$\hat{y} = \{\hat{y}_{\epsilon_1}, \ldots, \hat{y}_{\epsilon_n}\}$;
return $\hat{y}$

For multiclass classification, MC-AA is modified by assigning the class predictions for FGSM as provided in Algorithm 1. This modification takes into consideration the multiple classes where the given input is perturbed towards/away from the direction of its predicted class regardless of other classes. Referring to [2], the predictive mean can be computed as follows:

$$\hat{y} = p_{MC-AA}(y|x) \approx \frac{1}{T} \sum_{i=1}^{T} \hat{y}_{\epsilon_i}, \tag{2}$$

where $\widehat{y}_{\epsilon_i}$ is the output associated with x_{ϵ_i} at ϵ_i, and $i = 1, 2, \ldots, T$, where T is the total number of the produced outputs to be tuned.

To obtain the predictive uncertainty, we use mutual information (MI) that can be computed as follows:

$$\widehat{I}(y|x) = \widehat{H}(y|x) + \sum_c \frac{1}{T} \sum_{i=1}^{T} p(y = c|x_{\epsilon_i}) \log p(y = c|x_{\epsilon_i}), \tag{3}$$

where c is the class label, and

$$\widehat{H}(y|x) = -\sum_c p_{MC-AA}(y = c|x) \log p_{MC-AA}(y = c|x) \tag{4}$$

3.2 Monte-Carlo dropout: MC-dropout

In this method, the dropout is activated during the testing phase which is applied after each weight layer in a neural network. Given a neural network with input x and its observation y that is trained on $\mathcal{D}_{train}$, with L layers and learnable parameters w, then the predictive distribution can be written as:

$$p(y|x, \mathcal{D}_{train}) = \int p(y|x, w) p(w|\mathcal{D}_{train}) dw, \tag{5}$$

where $p(y|\ x,\ w)$ is the likelihood of the model and $p(w|\ D_{train})$ is the posterior distribution over the weights. Referring to [13], the posterior distribution – which is intractable – can be approximated by $q(w)$ by the minimisation of Kullback-Leibler divergence. By variational inference, the approximated predictive distribution becomes:

$$q(y|x) = \int p(y|x, w) q(w) dw \tag{6}$$

Gal et al. [13] has chosen the approximated posterior $q(w)$ as the distribution over the matrix of learnable weights with the randomly dropped out connections for posterior approximation. This is performed using the dropout during the testing phase. In other words, $q(w)$ can be defined as:

$$W_i = M_i . diag\left(\left[z_{i,j} \right]_{j=1}^{K_i} \right), \tag{7}$$

with $z_{i,j}$ as the realisations drawn from Bernoulli distribution for $i = 1, \ldots, L$ and $j = 1, \ldots, K_{i-1}$ such that the size of matrix W_i is $K_i x K_{i-1}$.

By drawing T samples from Bernoulli distribution, this produces $\left\{ W_1^t, \ldots, W_L^t \right\}_{t=1}^{T}$ which so allows to express the approximated predictive mean of a given input as:

$$E_{q(y|x)(y)} \approx \frac{1}{T} \sum_{t=1}^{T} \widehat{y}\left(x, W_1^t, \ldots, W_L^t\right) = p_{MC}(y|x) \tag{8}$$

Hence the predictive uncertainty using mutual information can be expresses as:

$$\widehat{I}(y|x,\mathcal{D}_{train}) = \widehat{H}(y|x,\mathcal{D}_{train}) + \sum_{c}\frac{1}{T}\sum_{t=1}^{T} p(y=c|x,w)\log p(y=c|x,w) \tag{9}$$

Where c is the class label and

$$\widehat{H}(y|x,\mathcal{D}_{train}) = -\sum_{c} p_{MC}(y=c|x,w)\,\log p_{MC}(y=c|x,w) \tag{10}$$

3.3 Deterministic uncertainty quantification (DUQ)

DUQ consists of feature extractor as a base model followed by an additional learnable layer to obtain the feature vectors corresponding to each class. The predictions are performed by computing a kernel function. The kernel function is the RBF kernel which computes the distance between the feature vectors and the centroids. The centroid of each class is updated using an exponential moving average of the feature vectors of the data points corresponding to the class with a momentum γ. The predictive uncertainty is obtained in a single deterministic forward pass. The output of a DUQ model can be expressed, referring to [37], as:

$$K_c(f_\theta(x), e_c) = \exp\left(-\frac{\frac{1}{n}\|W_c f_\theta(x) - e_c\|_2^2}{2\sigma^2}\right) \tag{11}$$

Where f_θ is the feature extractor mapping from input x of dimension m to the feature vectors of dimension d, and learnable parameters θ. W_c – for a class c – is a weight matrix of size n by d corresponding to the additional layer that transforms the output of the feature extractor to new embedding space with centroids size. K_c – the kernel output – is computed for each centroid class e_c with σ being the hyperparameter called the length scale. The prediction of this model is represented as:

$$\text{argmax}_c\, K_c(f_\theta(x), e_c). \tag{12}$$

Hence, the predictive uncertainty can be obtained by finding: $\max_c K_c(f_\theta(x), e_c)$.

The optimisation function can be expressed as:

$$L(x,y) = -\sum_{c} y_c \log(K_c) + (1-y_c)\log(1-K_c) \tag{13}$$

Moreover, there is further regularisation using two-sided gradient penalty (l_2 norm) where this penalty consists of regularisation factor λ to be tuned.

3.4 Deep ensemble

Deep Ensemble is a collection of deep models with different initialisations. Training multiple models with distinct initialised weights produce multiple outputs on a given prediction like MC-dropout but with independent parameters. Hence, the predictive mean can be obtained as follows:

$$\hat{y} = p_{ensemble}(y = c|x) = \frac{1}{T}\sum_{i=1}^{T} M_i(x),$$

where M_i(x) is the prediction obtained by model M_i on a given input x. The predictive uncertainty is obtained using Eqs. 9 and 10 but replacing p_{MC} by $p_{ensemble}$.

4 Evaluating model uncertainty

To evaluate the goodness of model uncertainty, we follow the same procedure applied in [26]. The predictive mean beside the predictive uncertainty can reflect the model uncertainty. Predictive mean provides the correct or incorrect classification with respect to the actual labels. Predictive uncertainty is derived from MI measurement, in which an arbitrary threshold T_u is set to classify MI between certain and uncertain. By tying predictive mean with uncertainty, we can realise four states that resemble the binary classification task as provided in Table 1. For simplicity, MI measurements are normalised via min-max with respect to the test set. Consequently, T_u is an arbitrary threshold between 0 and 1. The following abbreviations TN, FN, FP, TP correspond to true negatives, false negatives, false positives, and true positives, respectively.

Referring to Table 1, higher TN and TP are desired, with lower FP and FN. However, FN hurts the goodness of uncertainty in which erroneous predictions with high certainty are produced. On the other hand, FP is preferably required to be low, but this does not affect the performance of model uncertainty because uncertain and correct examples can be forwarded to an annotator. These measurements can be written as conditional probabilities to assess the performance of model uncertainty as following:

- Accuracy of model uncertainty:

$$A_u = \frac{TN + TP}{TN + FN + FP + TP}$$

- Negative Predictive Value (NPV): This can be expressed as conditional probability:

$$p(correct|certain) = \frac{p(correct, certain)}{p(certain)} = \frac{TN}{TN + FN}$$

Table 1 Possible cases in uncertainty estimation model

Model Uncertainty	Certain: $MI < T_u$	Uncertain: $MI \geq T_u$
Correct	TN: Correct and Certain	FP: Correct and Uncertain
Incorrect	FN: Incorrect and Certain	TP: Incorrect and Uncertain

- True Positive Rate (TPR): This can be expressed as a conditional probability:

$$p(uncertain|incorrect) = \frac{p(uncertain, incorrect)}{p(incorrect)} = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR): It can be written as a conditional probability:

$$p(correct|uncertain) = \frac{p(correct, uncertain)}{p(uncertain)} = \frac{FP}{FP + TN}$$

Furthermore, the last two metrics can be used to plot Receiver-Operation-Curve (ROC) and compute Area-Under-Curve (AUC) score to evaluate the goodness of model uncertainty by moving the threshold T_u between 0 and 1.

5 Experiments and results

In our experiments, we apply different machine learning models on graph and image datasets known as Cora and MNIST, respectively. Then, we estimate uncertainties besides the predictions to evaluate and compare the different uncertainty estimation methods. We use Pytorch [31] and Pytorch-Geometric package [11] in Python programming language.

5.1 Experimenting with graph data

As an example of graph data, we use Cora dataset to assess the proposed uncertainty method. Cora is a graph-structured data that comprises academic publications as nodes, and citations as the edges [33]. This data is used in node classification tasks in which each node is classified into one of seven subjects. The node features reflect the absence/presence as 0/1 of the corresponding word in the dictionary, in which the unique words in the dictionary are the total number of features. The data is described in Table 2. In this paper, we follow the same experimental setup for this data as in [42].

Since Cora is graph-structured data, we choose various graph neural network models to perform node classification. The graph learning models are arbitrarily chosen as following:

- GCN: Graph Convolutional Network based spectral approach. GCN layer can be expressed as:

$$x_i^{'} = \Theta \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\widehat{d}_j \widehat{d}_i}} x_j$$

- GraphConv: Graph Convolutional Network based spatial approach. GraphConv layer can be expressed as:

Table 2 Graph network description of Cora data

Cora Data	
# Nodes	2708
# Edges	5429
# Classes	7
# Features	1433
Train/Validation/Test	140/500/1000 nodes

$$x_i^{'} = \Theta_1 . x_i + \Theta_2 \sum_{j \in \mathcal{N}(i)} e_{j,i} . x_j$$

- GAT: Graph Attention Network. GAT layer can be expressed as:

$$x_i^{'} = \alpha_{i,i} \Theta . x_i + \Theta \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} . x_j$$

- SAGEConv: Graph SAGE Convolution. It is expressed as:

$$x_i^{'} = \Theta_1 . x_i + \Theta_2 . \text{mean}_{j \in \mathcal{N}(i)}\, x_j$$

- LEConv: Local Extremum Convolution.

$$x_i^{'} = \Theta_1 . x_i + \sum_{j \in \mathcal{N}(i)} e_{j,i} . \left(\Theta_2 . x_i - \Theta_3 . x_j \right)$$

- TAGConv: Topology Adaptive GCN. It is written as:

$$x_i^{'} = \sum_{k=0}^{K} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \Theta_k \left(\frac{e_{j,i}}{\sqrt{\widehat{d}_j \, \widehat{d}_i}} \right)^k x_j$$

where $x_i^{'}$ is the embedding derived from the input node i in the hidden layer, Θ_k is the learnable weight matrix at layer k, $e_{i,j}$ is the edge weight which is arbitrarily equal to 1, **mean** is the average over the sum, $\widehat{d}_i$ is the degree of node i and $\mathcal{N}(i)$ is the set of nodes in neighbourhood of node i.

The widths of all hidden layers are set to 16 neurons, a dropout after each hidden layer is set to 0.5 and the number of epochs is set to 100. We use a non-weighted NLLLoss and Adam optimiser to train the given models. Each of the preceded models consists of two graph convolutional layers. All hidden layers are squashed by ReLU and the output layers are followed by softmax function except for DUQ that uses RBF kernel as output.

Table 3 Tuned hyper-parameters of uncertainty methods using Cora Data

Model Uncertainty	Hyper-parameters (Cora)
MC-AA	ϵ_{max}=0.15, T=10, $\beta = \frac{2\epsilon_{max}}{10}$
MC-dropout	T=10, dropout=0.5
DUQ	$\lambda = 1x10^{-2}$, σ=0.3, γ=0.99
Deep Ensemble	T=10

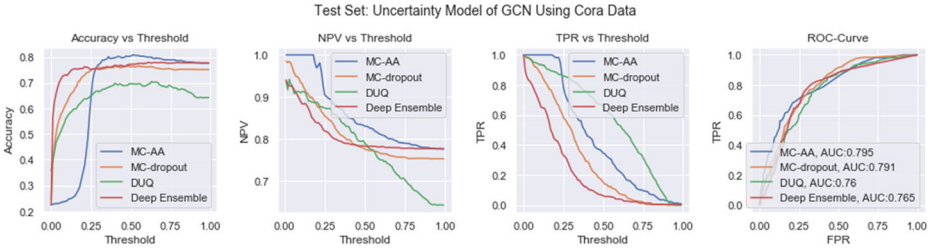

Fig. 2 GCN model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

To capture model uncertainty, we use MC-AA, MC-dropout, DUQ and Deep Ensemble methods. The hyper-parameters for these methods are empirically tuned which are summarised in Table 3.

After computing MI, we plot A_u, NPV, TPR and ROC to compare our proposed method with MC-dropout as depicted in Figs. 2, 3, 4, 5, 6 and 7.

5.2 Experimenting with image data

Regarding MNIST dataset [21], the data is divided as 90 k/10 k for train/test split. We use convolutional neural network (CNN) model that appeared in [37] as a deep feature extractor. The deep model consists of 3 CNN layers with output channels 64, 128 and 128, respectively, and a feed-forward layer with widths of 256. The kernel size is set to 3. Moreover, batch normalisation after every convolutional layer followed by 2 max pooling of size 2 by 2 is applied. The padding in the first two convolutional layer is set to 1. A dropout of value 0.5 is empirically set after the feed-forward hidden layer. The learning rate is set to 0.001, chosen empirically. The output layer is followed by softmax function to output the class prediction which is one of the handwritten digits from 0 to 9. The batch size is arbitrarily chosen to be 1024, then we perform 30 epochs to train the model. This model has attained an accuracy over 98% to classify the digits. Likewise, we capture model uncertainty on CNN model by performing MC-AA, MC-dropout, DUQ, and Deep Ensemble where the hyper-parameters are summarised in Table 4. As the input images 28 × 28 are grey-scale, the perturbed inputs by MC-AA should be clamped between −1 and 1 which is the range of the pixel values. We plot the preceded model uncertainty metrics as depicted in Fig. 8.

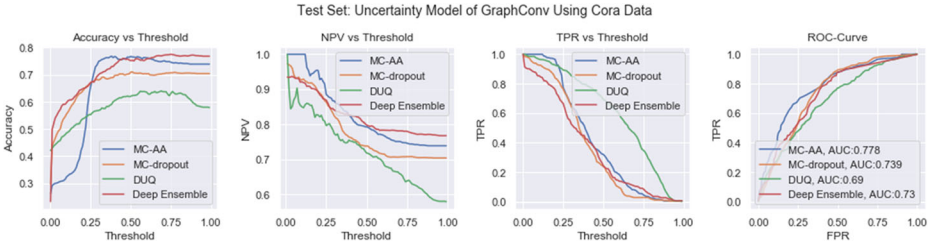

Fig. 3 GraphConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

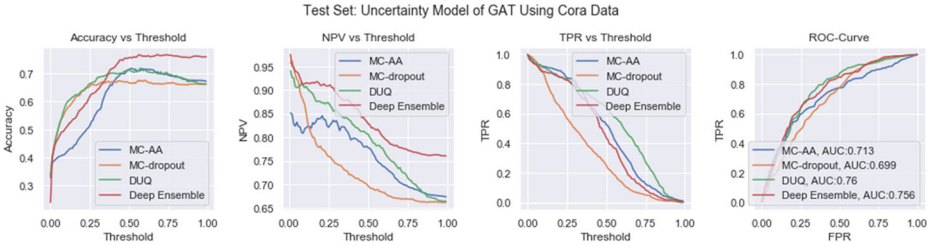

Fig. 4 GAT model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

6 Discussion

6.1 Uncertainty performance with Cora data

The performance of model uncertainty is computed by moving a threshold between 0 and 1 and computing the evaluation metrics provided earlier at each threshold. Generally, MC-AA for multiclass classification, has noticeably revealed competence against other uncertainty estimation methods on various graph learning models with Cora data in terms of accuracy and ROC-AUC curve plots in Figs. 2, 3, 4, 5, 6 and 7. Regarding other measurements, NPV metric has shown the highest with MC-AA against other methods with GCN and TAGCN models. TPR metric has revealed acceptable outcomes with MC-AA in the different graph learning models except for LEConv. All graph learning models have admitted the outperformance of DUQ in the subplots corresponding to TPR metrics. The same models have revealed the poor accuracy using DUQ model wherein the overall model is considered with a deficient performance. Despite MC-AA has revealed competent uncertainty estimates, this method has performed poorly with LEConv model. The reason of the deficient performance is due to multiple class distributions that restrict the behaviour of MC-AA. In other words, FGSM method in multiclass models might produce a perturbed input that cannot escape its relevant class. Thus, the perturbation using adversarial attack idea is not exact towards the decision boundary as we use the sign of gradients which is the L_∞ norm and neglect the ratios corresponding to each dimensional feature. Therefore, the perturbation on the given input does not allow this data point to fall in another class.

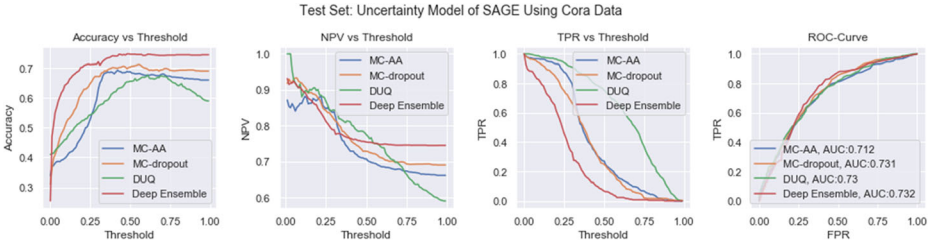

Fig. 5 SAGEConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

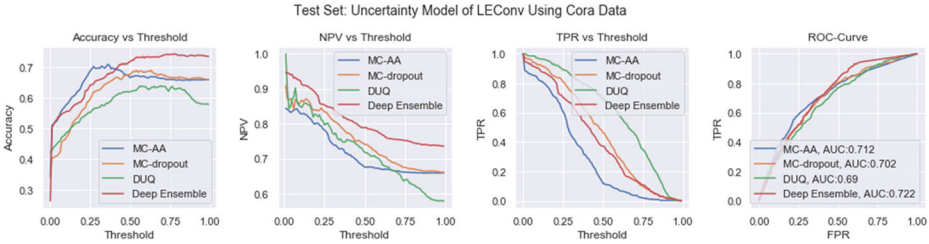

Fig. 6 LEConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

For this reason, we propose a way to avoid this drawback of MC-AA by converting multi-class classification problems into one-versus-all binary classification which has appeared to be more effective with MC-AA. We choose LEConv which performed poorly with MC-AA. To convert to "one-versus-all" classification, we choose the class with the highest false instances among all other classes to be the positive class, which is class 4, while the remaining labels are assigned as the negative class. However, the same concept can also be applied to the different permutations of one-versus-all binary classifications (e.g., first class versus others, second class vs others, etc.). The model LEConv is trained following the same experimental setup as preceded. Henceforth, we capture model uncertainty using MC-AA (for binary classification), MC-dropout, DUQ, and Deep Ensemble. The results are provided in Fig. 9.

After converting the classes of Cora data into binary labels, MC-AA has shown a superior success against other uncertainty methods. Clearly, low FN (incorrect and certain) is provided. Here, the sign of the gradients in FGSM method allows the data points to jump to the opposite class as there are only two competing classes.

6.2 Uncertainty performance with MNIST data

Referring to Fig. 8, the model uncertainty of CNN using MC-AA has outperformed other uncertainty methods using MNIST data. The overall model performance among all methods have attained the same accuracy. Whereas NPV and TPR metrics have depicted superior success with MC-AA wherein lower FN (incorrect and certain) has been obtained. To highlight the effectiveness of uncertainty estimates, we plot the normalised density distribution of the predictive uncertainties in Fig. 10. We cluster the distributions according to the correct/incorrect predictions, referring to Table 1. Clearly, all methods

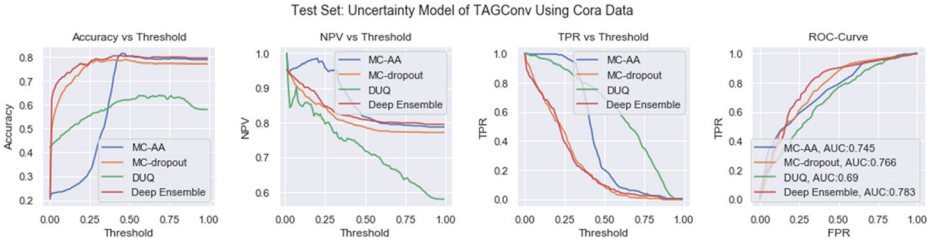

Fig. 7 TAGConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

Table 4 Tuned hyper-parameters of uncertainty methods using MNIST Data

Model Uncertainty	Hyper-parameters (MNIST)
MC-AA	ϵ_{max}=0.2, T=10, $\beta = \frac{2\epsilon_{max}}{10}$
MC-dropout	T=10, dropout=0.5
DUQ	λ=1$x10^{-1}$, σ=0.4, γ=0.9
Deep Ensemble	T=10

commonly have reflected good uncertainty estimates for correct predictions. However, the distribution of incorrect predictions among all uncertainty methods has shown different densities. The density of incorrect predictions with MC-AA is more concentrated towards the right values of mutual information where these predictions are considered uncertain. With other methods, the predictive uncertainty of incorrect predictions is distributed among the whole mutual information scale. Moreover, the uncertainty estimates with DUQ model have depicted more mix between correct/incorrect densities which is reflected in Fig. 9. In addition, we compute the mean/standard deviation to describe these distributions as provided in Table 5. The mean of incorrect predictions with MC-AA has attained the highest value with the smallest standard deviation. This is desired to obtain an effective model uncertainty that classifies incorrect predictions as uncertain. On the other hand, all methods have shown adequate results of low mean/low standard deviation for correct predictions. To pinpoint the effectiveness of MC-AA on the image MNIST data, we provide a case-study where we opt for an image example, depicted in Fig. 1, that is wrongly predicted by CNN among all methods. We investigate the predicted class of this digit two image as well as its uncertainty estimates by MC-AA, MC-dropout, DUQ and Deep Ensemble as provided in Fig. 11. Closely, all methods have erroneously predicted this digit as seven with high confidence, whereas the class two has provided a low predicted probability.

7 Conclusion

We have extended the study of MC-AA, an uncertainty estimation method based adversarial attack that works for multiclass classification. By benchmarking MC-AA method against other uncertainty estimation methods, we have shown the effectiveness of MC-AA in capturing model uncertainty using deep models on graph data of Cora and image data of MNIST. Concisely, we have examined MC-AA with multiclassification

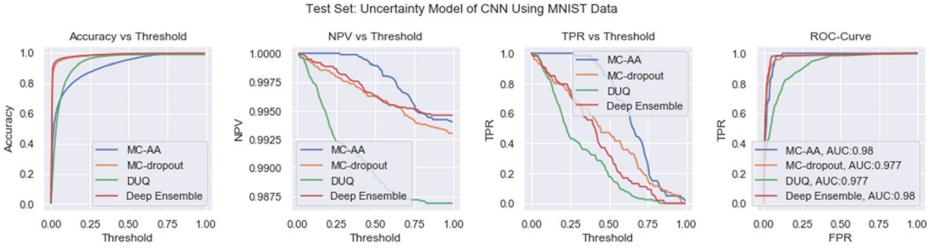

Fig. 8 CNN model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

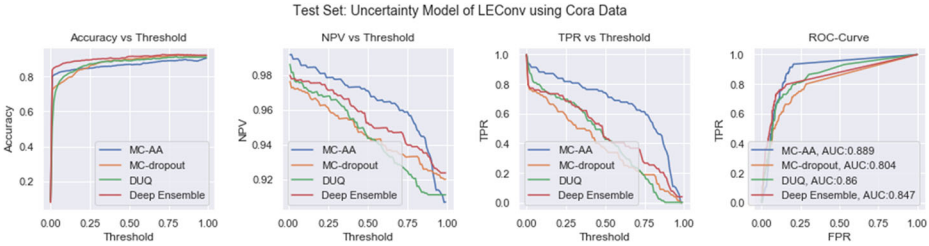

Fig. 9 Model uncertainty of LEConv as binary classification (class 4 vs rest) using MC-AA and MC-dropout. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u

tasks against MC-dropout, DUQ and Deep Ensemble method. However, the presented method has revealed low performance in LEConv model using Cora data. We tackle this limitation by converting the multiclass classification into binary classification task where MC-AA outperforms other methods. We discuss the best uncertainty model performance of MC-AA which attained an AUC score of 0.889 with LEConv model, after converting the Cora data to binary classification, where the corresponding NPV and TPR have also revealed superior success in comparison to other methods. Surprisingly, MC-AA has shown a significant outperformance using MNIST without the need to convert to binary classification. The recorded AUC-score of this method is 0.98 outperforming other methods in addition to their NPV and TPR curves. To wrap up, MC-AA is powerful in reducing the number of false negatives of model uncertainty (i.e., data points that are incorrect but certain). This is due to the perturbations that are performed on the input level, unlike previous uncertainty methods. The limitation of this study is that it is not

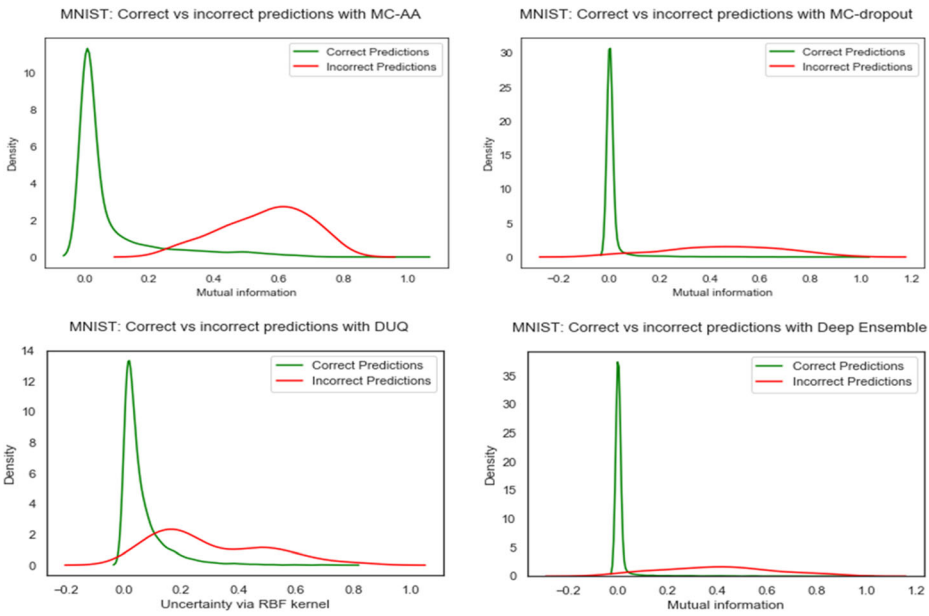

Fig. 10 Distribution of predictive uncertainty measurements derived from MNIST test set using MC-AA, MC-dropout, DUQ and Deep Ensemble. Correct predictions curve is the uncertainty measurements of the data points that are predicted correctly. Similarly, incorrect predictions curve is the uncertainty measurements of the data points that are predicted incorrectly

Table 5 Descriptive statistics summary of predictive uncertainty distribution using MNIST test set

Uncertainty method	Predictive uncertainty density	Mean	Standard deviation
MC-AA	Correct prediction	0.099	0.177
	Incorrect prediction	0.765	0.144
Mc-dropout	Correct prediction	0.016	0.069
	Incorrect prediction	0.456	0.22
DUQ	Correct prediction	0.073	0.085
	Incorrect prediction	0.272	0.1753
Deep Ensemble	Correct prediction	0.012	0.058
	Incorrect prediction	0.398	0.218

known when MC-AA is a good approach in multiclassification tasks. However, the conversion to binary classification is always promising since MC-AA tends to perturb an input between decision boundaries. Consequently, having a single decision boundary leads to effective results. Whereas multiple classes mislead the perturbed input which fails to reflect a good uncertainty estimate using FGSM sign method in MC-AA.

We foresee in future work to replace FGSM that uses l_∞ to more effective norm in MC-AA, where the gradients directions are more accurate towards the class boundary. This could lead to accurate perturbation towards the class boundary and eventually produce better uncertainty estimates.

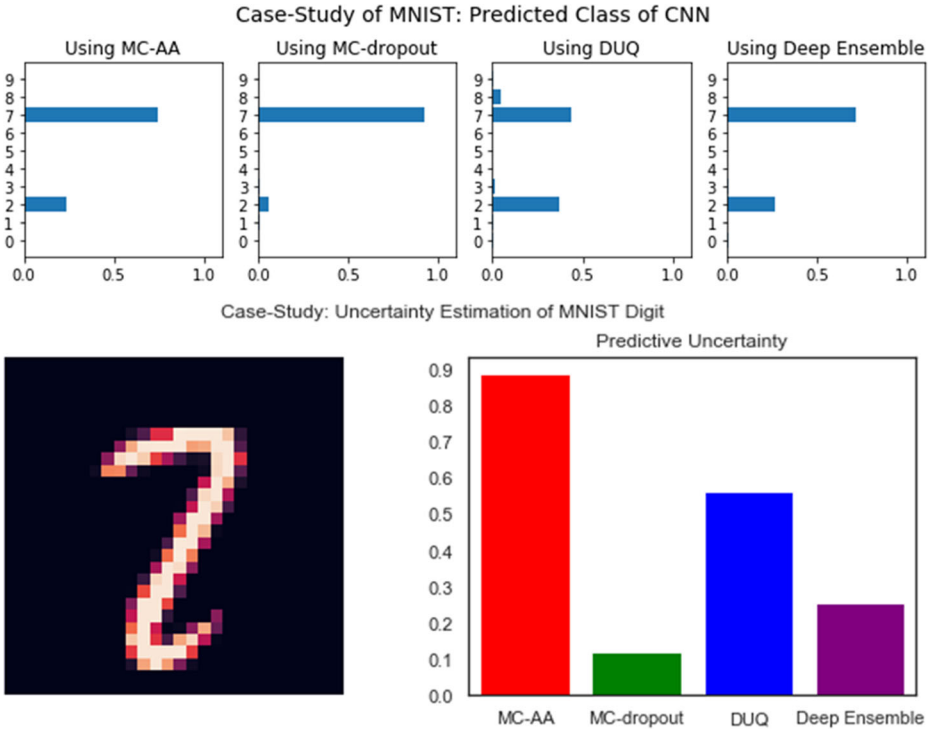

Fig. 11 Case study on MNIST test set. The top subplot is the predictive probability by the various CNN models (CNN with MC-AA, MC-dropout, DUQ, Deep Ensemble) of a single MNIST digit shown to the bottom left subplot. The bottom right subplot belongs to its uncertainty estimate derived from different uncertainty estimation methods

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abdar M, Pourpanah F, Hussain S, Rezazadegan D, Liu L, Ghavamzadeh M, Fieguth P, Cao X, Khosravi A, Acharya UR et al (2021) A review of uncertainty quantification in deep learning: techniques, applications and challenges, information fusion
2. Alarab I, Prakoonwit S (2021) Adversarial attack for uncertainty estimation: identifying critical regions in neural networks. Neural Process Lett:1–17
3. Alarab I, Prakoonwit S, Nacer MI (2020) Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In: Proceedings of the 2020 5th International Conference on Machine Learning Technologies, pp 23–27
4. Alarab I, Prakoonwit S, Nacer MI (2020) Comparative analysis using supervised learning methods for anti-money laundering in bitcoin. In: Proceedings of the 2020 5th International Conference on Machine Learning Technologies, pp 11–17
5. Alarab I, Prakoonwit S, Nacer MI (2021) Illustrative discussion of mc-dropout in general dataset: uncertainty estimation in bitcoin. Neural Process Lett 53(2):1001–1011
6. Ambati LS, El-Gayar O (2021) Human activity recognition: a comparison of machine learning approaches. J Midwest Assoc Inf Syst 2021(1):49
7. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural network. In: International conference on machine learning. PMLR, pp 1613–1622
8. Chakraborty A, Alam M, Dey V, Chattopadhyay A, Mukhopadhyay D (2018) Adversarial attacks and defences: a survey, arXiv preprint arXiv:1810.00069
9. Cuzzocrea A, Fadda E, Mumolo E (2022) Cyber-attack detection via non-linear prediction of ip addresses: an innovative big data analytics approach. Multimed Tools Appl 81(1):171–189. https://doi.org/10.1007/s11042-021-11390-1
10. El-Gayar OF, Ambati LS, Nawar N (2020) Wearables, artificial intelligence, and the future of healthcare. In: AI and big data's potential for disruptive innovation. IGI Global, pp 104–129
11. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch geometric. In: ICLR workshop on representation learning on graphs and manifolds
12. Gal Y (2016) Uncertainty in deep learning. University of Cambridge 1 (3), pp 4
13. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International conference on machine learning. PMLR, pp 1050–1059
14. Gal Y, Hron J, Kendall A (2017) Concrete dropout. arXiv preprint arXiv:1705.07832
15. Graves A (2011) Practical variational inference for neural networks. In: Advances in neural information processing systems. Citeseer, pp 2348–2356
16. Handa A, Sharma A, Shukla SK (2019) Machine learning in cybersecurity: a review. Wiley Interdiscip Rev Data Min Knowl Discov 9(4):e1306
17. Haq IU, Du X, Jan H et al (2022) Multimed Tools Appl. https://doi.org/10.1007/s11042-022-13154-x
18. Hernández-Lobato JM, Adams R (2015) Probabilistic backpropagation for scalable learning of bayesian neural networks. In: International conference on machine learning. PMLR, pp 1861–1869
19. Kendall A, Gal Y (2017) What uncertainties do we need in bayesian deep learning for computer vision? Adv Neural Inf Proces Syst 30
20. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. Adv Neural Inf Proces Syst 30

21. LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L (1989) Handwritten digit recognition with a back-propagation network. Adv Neural Inf Proces Syst 2
22. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
23. Lee H, Han S, Lee J (2017) Generative adversarial trainer: Defense to adversarial perturbations with gan, arXiv preprint arXiv:1705.03387
24. MacKay DJ (1992) A practical bayesian framework for backpropagation networks. Neural Comput 4(3): 448–472
25. Michelmore R, Kwiatkowska M, Gal Y (2018) Evaluating uncertainty quantification in end-to-end autonomous driving control, arXiv preprint arXiv:1811.06817
26. Mobiny A, Nguyen HV, Moulik S, Garg N, Wu CC (2019) Dropconnect is effective in modeling uncertainty of bayesian deep networks, arXiv preprint arXiv:1906.04569
27. Mobiny A, Yuan P, Moulik SK, Garg N, Wu CC, Van Nguyen H (2021) Dropconnect is effective in modeling uncertainty of bayesian deep networks. Sci Rep 11(1):1–14
28. Neal RM (1992) Bayesian training of backpropagation networks by the hybrid Monte Carlo method, Tech. rep., Citeseer
29. Neal R (1995) Bayesian learning for neural networks [phd thesis], Toronto, Ontario, Canada: Department of Computer Science, University of Toronto
30. Ovadia Y, Fertig E, Ren J, Nado Z, Sculley D, Nowozin S, Dillon J, Lakshminarayanan B, Snoek J (2019) Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. Adv Neural Inf Proces Syst 32
31. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alch e-Buc F, Fox E, Garnett R (eds), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., pp 8024–8035. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
32. Pérez-Gil Ó, Barea R, López-Guillén E, Bergasa LM, Huélamo CG, Gutiérrez R, Díaz-Díaz A (2022) Deep reinforcement learning based control for autonomous vehicles in carla. Multimed Tools Appl 81(3):3553–3576. https://doi.org/10.1007/s11042-021-11437-3
33. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–93
34. Shen S, Jin G, Gao K, Zhang Y (2017) Ape-gan: Adversarial perturbation elimination with gan, arXiv preprint arXiv:1707.05474
35. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
36. Tynan D, Yadron D Tesla driver dies in first fatal crash while using autopilot mode, https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk. Accessed 16 July 2021
37. Van Amersfoort J, Smith L, Teh YW, Gal Y (2020) Uncertainty estimation using a single deep deterministic neural network. In: International Conference on Machine Learning. PMLR, pp 9690–9700
38. van Amersfoort J, Smith L, Jesson A, Key O, Gal Y (2021) Improving deterministic uncertainty estimation in deep learning for classification and regression, arXiv preprint arXiv:2102.11409
39. Wang S, Wang X, Zhao P, Wen W, Kaeli D, Chin P, Lin X (2018) Defensive dropout for hardening deep neural networks under adversarial attacks. In: Proceedings of the International Conference on Computer-Aided Design, pp 1–8
40. Wang C, Wang X, Zhang J, Zhang L, Bai X, Ning X, Zhou J, Hancock E (2022) Uncertainty estimation for stereo matching based on evidential deep learning. Pattern Recogn 124:108498
41. Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robinson T, Leiserson CE (2019) Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics, arXiv preprint arXiv:1908.02591
42. Yang Z, Cohen W, Salakhudinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: International conference on machine learning, PMLR, pp 40–48