Soft Computing
https://doi.org/10.1007/s00500-023-08031-0

DATA ANALYTICS AND MACHINE LEARNING

Uncertainty estimation-based adversarial attacks: a viable approach for graph neural networks

Ismail Alarab[1] · Simant Prakoonwit[1]

Accepted: 10 March 2023

Abstract
Uncertainty estimation has received momentous consideration in applied machine learning to capture model uncertainty. For instance, the Monte-Carlo dropout method (MC-dropout), an approximated Bayesian approach, has gained intensive attention in producing model uncertainty due to its simplicity and efficiency. However, MC-dropout has revealed shortcomings in capturing erroneous predictions lying in the overlapping classes. Such predictions underlie noisy data points that can neither be reduced by more training data nor detected by model uncertainty. On the other hand, Monte-Carlo based on adversarial attacks (MC-AA), an outstanding method, performs perturbations on the inputs using the adversarial attack idea to capture model uncertainty. This method admittedly mitigates the shortcomings of the previous methods by capturing wrong labels in overlapping regions. Motivated by this method that was only validated with neural networks, we sought to apply MC-AA on various graph neural network models to obtain uncertainties using two public real-world graph datasets known as Elliptic and GitHub. First, we perform binary node classifications, then we apply MC-AA and other recent uncertainty estimation methods to capture the uncertainty of the models. Uncertainty evaluation metrics are computed to evaluate and compare the performance of the uncertainty of the model. We highlight the efficacy of MC-AA in capturing uncertainties in graph neural networks wherein MC-AA outperforms other given methods.

Keywords Uncertainty estimation · Adversarial attack · Graph neural network

1 Introduction

Graph neural networks (GNNs) have attained significant success in machine learning applications with the resurgence of graph data in several fields such as in bioinformatics (Lim et al. 2019; Harada et al. 2020), social networks (Xu et al. 2018; Rozemberczki et al. 2019; Guo and Wang 2020) and blockchain (Weber et al. 2019; Alarab et al. 2020a) to provide automated decision-making in complex structured data. However, graph learning models produce overconfident predictions that are not trustworthy. To obtain reliable predictions, the uncertainty of the model is paramount to address this issue. On the other hand, recent years have witnessed a surge of interest in uncertainty estimation-based Bayesian approximations to capture model uncertainty in machine learning models (Abdar et al. 2021). Hence, not only the predictions of the model assist in decision-making but also the quantified uncertainty. In classification tasks, uncertainty estimates can be discriminated between two different scenarios. The first scenario involves the data points falling out of distribution and they express the epistemic uncertainty, whereas the data points that fall near or at the decision boundary are known as aleatoric uncertainty.

Many studies have been conducted to reflect these scenarios using several uncertainty estimation methods as in Abdar et al. (2021); Gal and Ghahramani 2016; Amersfoort et al. 2021). For instance, Monte-Carlo dropout (MC-dropout), introduced in Gal and Ghahramani (2016) as a Bayesian approximation, performs multiple stochastic forward passes in the neural network with activated drop-out during the test phase to obtain model uncertainty. This method is capable of targeting data points that fall near the decision boundary of class distributions (Alarab et al.

✉ Ismail Alarab
ismail.arab.eng@gmail.com

Simant Prakoonwit
sprakoonwit@bournemouth.ac.uk

[1] Bournemouth University, Bournemouth, UK

2021), where this method has revealed promising and effective results in previous studies (Gal et al. 1705; Kennamer et al. 2019; Ng et al. 2018). Although its simplicity and efficiency produce uncertainties, MC-dropout has revealed a remarkable drawback. The drawback lies in the hindrance of this method in capturing data points that fall into overlapping classes. Henceforth, these data points produce certain erroneous predictions. Another method known as deterministic uncertainty quantification (DUQ) (Amersfoort et al. 2020) has also appeared to capture out-of-distribution data points using a single feedforward pass. However, this method captures only the data points which reside far from the training data.

The recent work in Alarab and Prakoonwit (2021) has proposed a novel method to estimate uncertainty based on an adversarial attack idea known as Monte-Carlo adversarial attack (MC-AA). Instead of perturbing the decision boundary as in MC-dropout, this method performs a direct perturbation to the input using the fast gradient sign method (FGSM). The MC-AA method reveals a significant effect in capturing data points between the overlapping classes at the cost of increasing the number of uncertain predictions with correct classifications that do not affect the uncertainty performance. Although the efficacy of MC-AA, this method has been merely proposed on multilayer perceptron (MLP) for binary classification (Alarab and Prakoonwit 2021).

Unlike the existing studies, our novelty in this paper is to provide a comprehensive study on the validity of MC-AA with the emergent graph neural network models for binary node classification tasks to produce uncertainty estimates besides the model's predictions. The main challenge is to examine the viability of the promising method MC-AA on two datasets called Elliptic and GitHub graph data from two distinct fields known as blockchain and social networks, respectively. These datasets are arbitrarily chosen but with binary labels to target binary node classification tasks. Our study has admittedly revealed the outperformance of MC-AA against other given methods wherein the former improves the fail-safes in graph neural networks.

This paper is structured as follows: Sect. 2 provides an overview of the related work. Section 3 and Sect. 4 demonstrate the uncertainty methods and the evaluation metrics, respectively. The experiments are provided in Sect. 5. Section 6 discusses the provided results, and a conclusion is presented in Sect. 7.

2 Overview of related works

The prominence of Bayesian approaches has emerged for several years in the estimation of the uncertainty of neural networks, known as Bayesian neural networks (BNNs) (MacKay 1992). BNNs have introduced the notion of priors over the weights of the neural network to produce posterior distributions (Neal 2012). Although BNNs are mathematically easy to formulate, their exact inference is intractable (Gal 2016). Also, Gaussian processes (GP) derived from Bayesian approaches have involved priors over the functions of neural networks instead of their weights (Rasmussen 2003). GPs have played an important role in the estimation of uncertainty throughout the years; however, they are prohibitively expensive (Gal 2016). Recently, studies in uncertainty estimation have come out with approximate Bayesian methods that are practically feasible as in Gal and Ghahramani (2016); Amersfoort et al. 2021; Amersfoort et al. 2020) (please refer to Abdar et al. (2021) for a comprehensive review of uncertainty estimation methods). Interestingly, the work in Gal and Ghahramani (2016) has provided a simple, efficient, and scalable method, MC-dropout, to capture model uncertainty. The idea of this method lies in providing multiple stochastic forward passes, during the testing phase, that samples priors from Bernoulli distributions as a Bayesian approximation. Practically, this method performs multiple perturbations on the decision boundary leading to an ensemble of decision functions to output the final predictions rather than uncertainty estimates. This method has shown its efficacy in producing an uncertainty estimation, especially data points near the decision boundary of the classifier (Alarab et al. 2021). The DUQ method (Amersfoort et al. 2020) is another uncertainty estimation method which captures the out-of-distribution data points where the lack of training data exists.

On the other hand, classification tasks are inherently subjected to noisy instances that cannot be reduced by acquiring more data. As a result, this leads to a region of overlapping classes. The limitation of MC-dropout and DUQ methods lies in capturing this type of data point. Since MC-dropout performs multiple perturbations on the decision boundary as an ensemble of decision functions, none of these functions can provide the correct label of a noisy point in the overlapping classes region. In addition, the DUQ model does not target this type of uncertainty. Henceforth, such instances are erroneously predicted with certainty.

To mitigate this issue, recent work in Alarab and Prakoonwit (2021) has proposed an uncertainty estimation method that uses the idea of adversarial attacks. This method, known as Monte-Carlo adversarial attack (MC-AA), applies back-and-forth perturbations in the direction of the decision boundary on each data point. The constructed perturbations are derived from the FGSM method which was originally introduced to calculate adversaries for a robust neural network classifier (Chakraborty et al. 2018). Although the promising performance of MC-AA, this method has been only applied and validated with multilayer perceptron (MLP) on tabular datasets. On the other hand, graph neural networks have witnessed growth and success in the machine learning era in recent years with the increase of graph data (Xu et al. 2018). For this purpose, we test and validate the efficacy of MC-AA on several graph learning algorithms, unlike any previous study, with graph datasets from different fields then we evaluate and compare MC-AA against other methods wherein promising results are provided.

This paper highlights the efficacy of MC-AA in the prominent GNN models to compute uncertainty estimation with the increase of graph datasets.

3 Uncertainty estimation methods

3.1 MC-AA

MC-AA stems from the idea of adversarial attacks. In white-box attacks, adversaries are deterministic noisy perturbations of inputs to fool the decision of a trained model (Chakraborty et al. 2018), wherein these attacks have a great impact on the safety of the model. The perturbed inputs can be obtained using the FGSM method. Let y_i be the observed output that corresponds to the input x_i for $i \in \{1, \ldots, N\}$ with N being the number of observations. A neural network can be expressed as $f(.)$ with input x and prediction $\widehat{y} = f(x)$ accompanied by a set of learnable weights $w = \{W_1, \ldots, W_L\}$ where L is the number of layers. Also, we denote by $J(.,.)$, the loss function of the neural network model. Using the FGSM method, we can express the adversarial example of the input by:

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

where $x(\epsilon)$ is the perturbed input known as an adversarial example by ϵ which is a small value and ∇_x is the gradient with respect to the original input x.

Let $\epsilon \in I = \{-\epsilon_{\max}, \ldots, 0, \beta, \ldots, \epsilon_{\max}\}$ such that I is symmetric interval evenly spaced by β, and bounded by $\epsilon_{\max}$ where $\epsilon_{\max}$ is a tunable hyperparameter. The FGSM method requires a target label to produce the perturbed version of the original input, so the target label is arbitrarily assumed to be 0 in the whole paper.

Hence, each input x can produce multiple adversarial examples x_{ϵ_j} associated with $\widehat{y}_{\epsilon_j} = f(x_{\epsilon_j})$ at ϵ_j for $j \in \{1, ..., |I|\}$ where $|I|$ denotes the cardinality of I. In other words, each test sample induces multiple inputs derived from the back-and-forth perturbations of the initial sample in the direction of the decision boundary. By performing MC-AA, multiple outputs $\widehat{y}_{\epsilon_j}$ are obtained from multiple perturbed versions x_{ϵ_j} of each data point x as summarised in Algorithm 1, wherein these outputs are used to estimate uncertainty.

The main types of uncertainty can be categorised into epistemic and aleatoric uncertainty. The lack of training examples in a newly tested point indicates epistemic uncertainty, whereas the noisy instances are said to be aleatoric. In (Gal and Ghahramani 2016), this work has introduced different measurements of uncertainty estimates such as mutual information. Furthermore, the work in Smith and Gal (1803) has claimed that mutual information is used to capture epistemic uncertainty in the model, in which mutual information has been used to convey the amount of information required by the model's output. However, it is not fully true that mutual information captures only epistemic uncertainty. This is due to the overconfident predictions provided by neural network optimisation. Thus, some data points of aleatoric uncertainty are expected to be captured too. Meanwhile, we are interested in the points lying near the decision boundary and between the overlapping classes, which are suitable to detect using MI. In this paper, we follow the procedure used in Gal and Ghahramani (2016) where the predictive mean of an input data x using the MC-AA method can be written as:

$$p_{\text{MC-AA}}(y|x, w) = \frac{1}{T}\sum_{j=1}^{T} p(y|x_{\epsilon_j}, w) \approx \frac{1}{T}\sum_{j=1}^{T} \widehat{y}_{\epsilon_j}, \tag{2}$$

where $\widehat{y}_{\epsilon_j}$ is the output associated with x_{ϵ_j} at ϵ_j and T is equivalent to $|I|$.

Besides the predictive mean, the predictive uncertainty derived from mutual information measurement, due to Gal and Ghahramani (2016), can be written as:

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

where c is the class label, $p(y|x_{\epsilon_i})$ is equivalent to $\widehat{y}_{\epsilon_i}$, and

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

Algorithm 1 MC-AA

```
Require:
  • $f(.)$: is a neural network that maps a data sample $x$ to output $\hat{y}$.
  • $J(.,.)$: is a loss function as defined earlier.
  • Input:
      - I: is a discrete interval comprising a set of values of $\epsilon$ as mentioned earlier.
        $\epsilon_{max}$ is a tunable hyper-parameter.
      - $x$: is a test data sample.
      - $\tilde{y}$: is an arbitrary output assumed on the input which is required for FGSM
        to move back and forth toward the decision boundary. We assume that all
        test points are labelled $\tilde{y} = 0$.
  • Output:
      - $\hat{y}$: is a set of outputs for a single data point. $\hat{y} = \{\hat{y}_{\epsilon_j}\}_{j=1}^{T}$, where $\epsilon_i \in I$ and
        $T = |I|$.

Function

Compute the gradients of the loss function with respect to $x$ as:
grad = $\nabla_x J(x, \tilde{y})$
for all $\epsilon_i \in I$ do
        $x_{\epsilon_i} = x + \epsilon_i . sign(grad)$;
        $\hat{y}_{\epsilon_i} = f(x_{\epsilon_i})$;
        return $\hat{y}_{\epsilon_i}$
end for

$\hat{y} = \{\hat{y}_{\epsilon_j}\}_{j=1}^{T}$;
return $\hat{y}$
```

3.2 MC-dropout

This method is based on multiple stochastic forward passes during the testing phase, in which a dropout function is applied after each weight layer (Gal and Ghahramani 2016). For each given input, multiple distinct outputs are obtained by drawing T samples from the Bernoulli distribution under the activated dropout. The weights of the neural network at sample t can be written as:

$$p_{\text{MCdrop}}(y|x, w) = \frac{1}{T}\sum_{t=1}^{T} y(x, W_1^t, \ldots, W_L^t). \quad (5)$$

Like MC-AA, the mutual information can be obtained as follows:

$$\widehat{I}(y|x) = \widehat{H}(y|x) + \sum_c \frac{1}{T}\sum_{t=1}^{T} p(y = c|x, w)\log p(y = c|\,x, w), \quad (6)$$

where c is the class label, and

$$\widehat{H}(y|x) = -\sum_c p_{\text{MCdrop}}(y = c|x, w)\log p_{\text{Mcdrop}}(y = c|x, w). \quad (7)$$

3.3 Deterministic uncertainty quantification (DUQ)

DUQ is formed of a feature extractor followed by a kernel to perform predictions. The feature extractor is used to learn the feature vectors corresponding to each class. The kernel computes the distance between the feature vectors and the class centroids. The centroid is computed and updated using the exponential moving average of momentum γ on the feature vectors of a certain class. Referring to Amersfoort et al. (2020), the output of the DUQ model using a radial basis function kernel can be written as:

$$K_c(f_\theta(x), e_c) = \exp\left(-\frac{\frac{1}{n}||W_c f_\theta(x) - e_c||_2^2}{2\sigma^2}\right), \quad (8)$$

where f_θ is the feature extractor output that maps an input vector x, e_c is the centroid of class c, $||.||_2$ is the l_2-norm, and σ is the length scale. W_c is a weight matrix corresponding to class c of size d where this matrix transforms the output of the feature extractor to new embeddings of the centroid size. A further two-sided gradient penalty is added to this model since the deep learning models are prone to feature collapse (Amersfoort et al. 2020). The two-sided gradient penalty can be written as follows:

$$\lambda . \left[\left\| \nabla_x \sum_c K_c \right\|_2^2 - 1 \right], \tag{9}$$

where λ is a hyperparameter to be tuned.

4 Uncertainty evaluation

We use the uncertainty measurements to perform uncertainty evaluation as introduced in Mobiny et al. (1906). Every test sample comprises a correct/incorrect classification accompanied by certain/uncertain predictions. Correct/incorrect is assigned according to the ground truth of labels with the predictive mean. Certain/uncertain is set according to the mutual information measurement, whereas the predictive uncertainty is below/above an arbitrary threshold T_u is certain/uncertain, respectively. Assuming that the mutual information values are normalised by "min–max" over the test set, T_u can vary between 0 and 1. Hence, we can distinguish between four possible states as follows: correct and certain, incorrect and certain, correct and uncertain, and incorrect and uncertain as shown in Table 1 that are used to evaluate the performance of model uncertainty.

Referring to Table 1, the goodness of model uncertainty can be reflected using the following expressions:

- Negative Predictive Value (NPV): It is desirable to have a correct classification when the model is certain of its predictions. This can be written as conditional probability:

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

Table 1 Model uncertainty states. This table shows the four possible states of model uncertainty to classify every test sample. This evaluation resembles a binary classification task. MI corresponds to mutual information measurement

Model uncertainty	Certain: $MI < T_u$	Uncertain: $MI \geq T_u$
Correct	True negatives: TN	False positives: FP
Incorrect	False negatives: FN	True positives: TP

- True Positive Rate (TPR): It is desirable to receive uncertain predictions when the classification is incorrect. This can be expressed as a conditional probability:

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

- False Positive Rate (FPR): This ratio expresses the correct predictions when the model is uncertain. This ratio does not have an impact on the performance of model uncertainty, but it is computed to obtain the Receiver-Operation-Curve (ROC curve). It can be written as a conditional probability:

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

In addition, the uncertainty evaluations provide a similar approach to using the Area-Under-Curve (AUC) score and ROC by tweaking the uncertainty threshold T_u between 0 and 1. On the other hand, we highlight the differences between FP and FN here. FP does not have a high impact on the model performance since the uncertain and correct predictions can be forwarded for further decision-making. While FN has a great impact on the model uncertainty since it reflects the incorrect classifications with certain predictions.

5 Experiments

We perform binary node classification on two arbitrarily chosen graph data, called elliptic and GitHub datasets using several GNNs models. Then, we compute the model uncertainty using MC-AA against other methods.

5.1 Data description

5.1.1 Elliptic dataset

Elliptic dataset is a public graph of data derived from the Bitcoin blockchain (Weber et al. 2019; Alarab et al. 2020b). The graph incorporates nodes as transactions and edges as the flow of payments. The nodes are partially labelled between licit (e.g., miners) and illicit (e.g., theft, scam, etc.) transactions (txs). These data consist of 49 directed acyclic graphs (DAG) in which each time-stamped graph refers to its time slot when extracted. The necessary description of these graph data is given in Table 2. As this data comprises 166 features which are local features (LF)

Table 2 Graph description of Elliptic and GitHub datasets

Graph network description	Elliptic	GitHub
Directed	Yes	No
Temporal	Yes	No
# Nodes	203,769	37,700
# Edges	234,355	289,003
# Node features	166	128
Binary Node Labels	Licit/Illicit txs	Web/ML developer

(e.g., timestamp, number of inputs/outputs, transaction fees) and aggregated features (AF), we only use LF in our experiments which counts to 94. The train/test sets are chosen following the temporal split of these data in which the first 29 graphs (136,265 nodes) belong to the train set, the graphs from 30 to 34 correspond to the validation set, and the remaining graphs belong to the test set. Then, this dataset is followed by a standardisation step.

As an example of graph data, we use Elliptic data. It is a public graph of data derived from the Bitcoin blockchain (Alarab et al. 2020a, 2020b). The graph incorporates nodes as transactions and edges as the flow of payments. The nodes are partially labelled between licit (e.g., miners) and illicit (e.g., theft, scam) transactions. These data consist of 49 directed acyclic graphs (DAG) in which each time-stamped graph refers to its time slot when extracted. The necessary description of these graph data is given in Table 2. As these data comprise 166 features which are local features (LF) (e.g., timestamp, number of inputs/outputs, transaction fees) and aggregated features (AF), we only use LF in our experiments after features are standardised.

5.1.2 The GitHub dataset

The GitHub dataset is a large social network of GitHub developers. This undirected graph network was extracted from the public API in June 2019 (Rozemberczki et al. 2019; Github social network xxxx). The graph network consists of nodes as developers who have starred at least 10 repositories, accompanied by edges as mutual follower relationships between developers. The node features are acquired based on the location, repositories starred, employer and e-mail address. The nodes acquire binary labels, derived from the job title of each user, to predict whether the Github user is a web or a machine learning (ML) developer. The description of the GitHub dataset is summarised in Table 2. We arbitrarily opt for the 0.7/0.1/0.2 ratio for the train/validation/test split, which is followed by a standardisation step.

5.2 Graph neural networks (GNNs)

We conducted our experiments using several graph learning models to perform binary node classification on the given datasets. We use PyTorch (Paszke et al. 2019) and PyTorch-geometric package (Fey and Lenssen 2019) in Python programming language. Hence, we chose a set of popular graph learning models as follows:

- GCN (Kipf and Welling 2017): Graph Convolutional Network-based spectral approach. The GCN layer can be written as:

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

- GraphConv (Morris et al. 2020): Graph Convolutional Network-based spatial approach. It is expressed as:

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

- GAT (Vlickovic et al. 2018): Graph Attention Network. It is expressed as:

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

- SAGEConv (Hamilton et al. 2018): Graph SAGE Convolution. It is expressed as:

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

- LEConv (Ranjan et al. 2020): Local Extremum Convolution.

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

- TAGConv (Du et al 2018): Topology Adaptive GCN. It is expressed as:

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

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 1 here, $\alpha_{i,j}$ is the attention coefficient, mean is the average over the sum, $\hat{d}_i$ is the degree of node i and $\mathcal{N}(i)$ is the set of nodes in the neighbourhood to node i.

5.3 Experimental settings and implementations

For simplicity, the experimental setup for models applied in each of the experiments on the Elliptic and GitHub

Table 3 Hyperparameters used for all GNN models

Hyperparameters	All models
Input/Hidden layers size	100
Learning rate	0.01
Dropout	0.2
# Epochs	50
Activation function	ReLU
Output function	Softmax
Weighted loss function	NLLLoss

datasets is set equally. For the given graph learning algorithms, the models are formed of input and output layers, and they are trained in a full-batched fashion on both datasets in which we empirically chose the hyperparameters as provided in Table 3. The weight of the loss function is empirically set to 0.3/0.7 for Elliptic data only, to mitigate class imbalance.

The classification results of the standard GNN models Elliptic and GitHub datasets are provided in Tables 4 and 5, respectively.

5.4 Experimenting MC-AA with GNN models

5.4.1 Obtaining uncertainty estimates using Elliptic dataset

We perform uncertainty estimation using MC-AA and compare it against MC-dropout and DUQ methods using the given GNN models. Using MC-AA as in Algorithm 1, we use a non-weighted loss NLLLoss and assume that all test points belong to class 0, and we arbitrarily set β to $\frac{\epsilon_{\max}}{10}$. The evenly spaced interval I is chosen such that it is bounded by $\epsilon_{\max} = 0.09$ chosen empirically using the highest AUC-score of model uncertainty in the GCN model. For brevity, $\epsilon_{\max}$ is kept the same for all models. Hence, we perform multiple perturbations (equal to $\frac{\epsilon_{\max}}{\beta} = 10$) back and forth in the direction of the boundary decision. For every data point, we have 10 distinct output predictions that can be used to estimate uncertainty via mutual information measurement. To perform the MC-dropout method, we simply activate the dropout during the testing phase, and we arbitrarily perform 50 stochastic forward passes on each input (data point) to produce model uncertainty using mutual information measurements.

For the DUQ model, we empirically set the hyperparameters as follows: $\lambda = 0.1$, $\gamma = 0.9$ and $\sigma = 0.3$.

We evaluate the performance of model uncertainty for all models using the procedure proposed earlier in this paper after computing TN, FN, FP and TP on the tested samples. We plot the evaluation metrics (NPV, TPR, ROC curves) that reflect the performance of MC-AA, MC-dropout and DUQ as a function of an arbitrary uncertainty threshold T_u for all mentioned models as shown in Figs. 1, 2, 3, 4, 5, and 6.

5.4.2 Obtaining uncertainty estimates using GitHub dataset

Like the preceding procedure, we apply MC-AA and MC-dropout methods on the test set of the GitHub dataset to evaluate the model uncertainty of the given GNN models

Table 4 Classification results of graph models using the Elliptic dataset

Models using elliptic	% Accuracy	% Precision	% Recall	% F_1 Score
GCN	95.6	73.9	50.1	59.7
GraphConv	95.15	74.3	38.8	51
GAT	95.5	77.1	45.15	56.9
SAGEConv	95.5	76	45	56.5
LEConv	95.9	79.3	49.95	61.3
TAGConv	93.93	52.8	62.3	57.2

Table 5 Classification results of graph models using the GitHub dataset

Models using GitHub	% Accuracy	% Precision	% Recall	% F_1 score
GCN	87.26	82.65	63.28	71.68
GraphConv	81.87	61.92	74.79	67.75
GAT	86.97	77.1	80.29	71.68
SAGEConv	87.06	80.66	64.74	71.82
LEConv	82.05	64.29	66.4	65.33
TAGConv	87.4	80.58	66.56	72.9

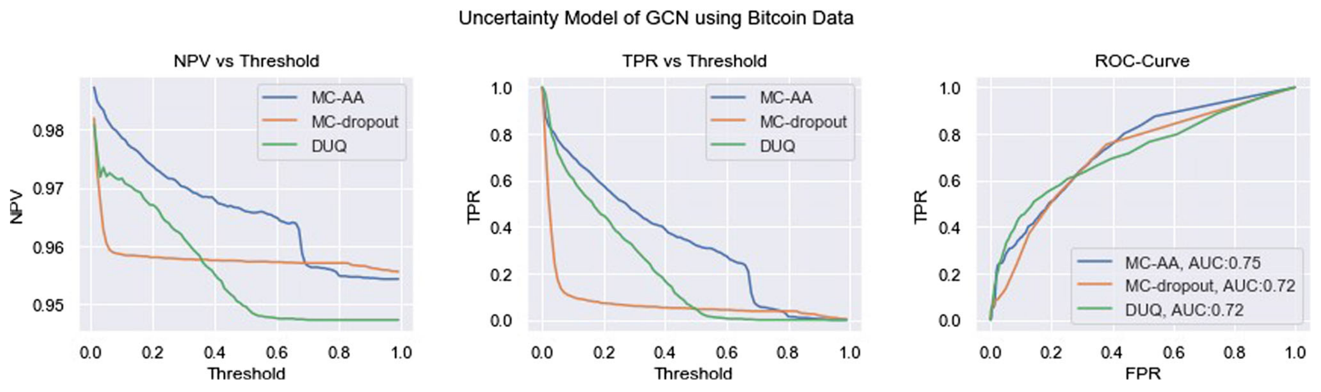

Fig. 1 Model uncertainty of GCN using the Elliptic dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

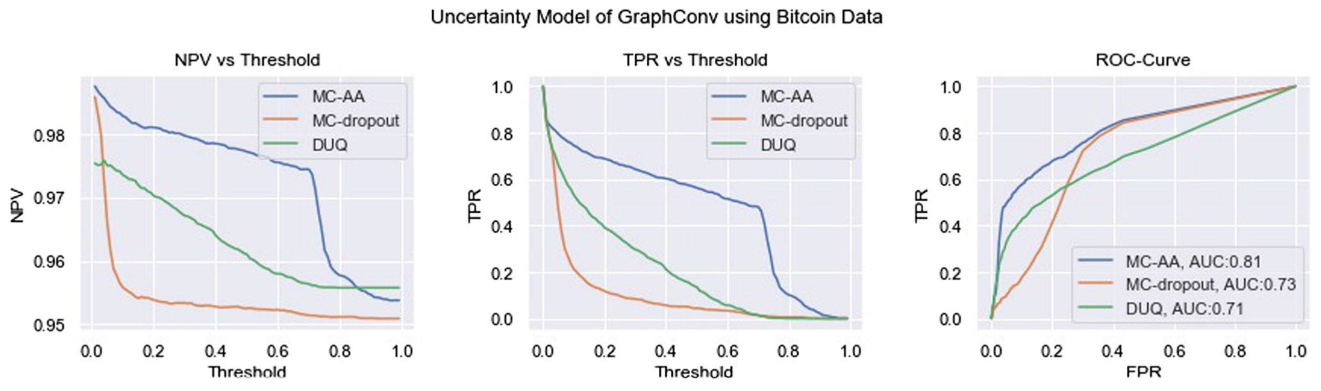

Fig. 2 Model uncertainty of GraphConv using the Elliptic dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

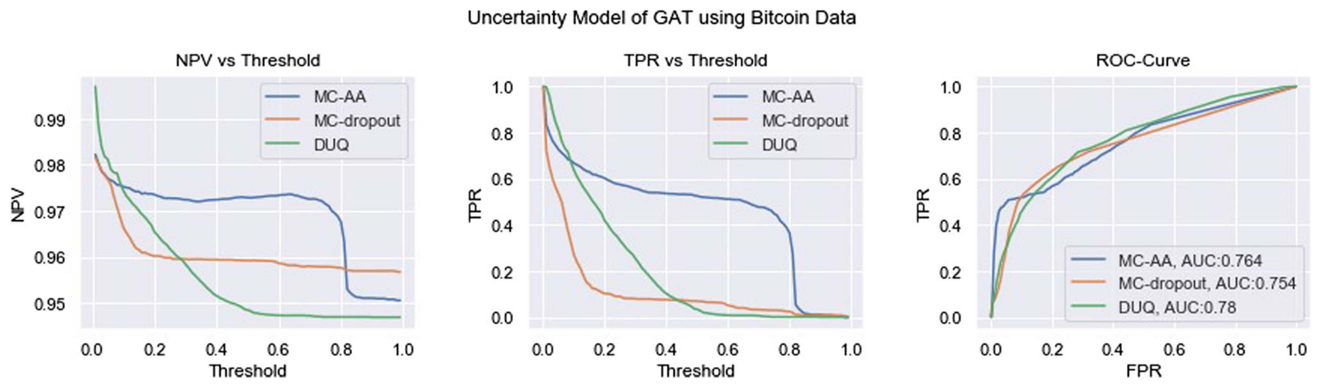

Fig. 3 Model uncertainty of GAT using the Elliptic dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

with the MC-AA method and compare it to that in the MC-dropout method. To perform MC-AA, we choose non-weighted NLLLoss with the test set arbitrarily assumed to belong to class 0 and $\beta = \frac{\epsilon_{\max}}{10}$ by default. Then, we empirically choose ϵ_{max} to be equal to 0.5 which has achieved the highest AUC score of model uncertainty on the GCN model. For simplicity, this hyperparameter is assigned equally in the rest of the GNN models. On the other hand, we perform MC-dropout with 50 stochastic forward passes on the test set.

The DUQ model is assigned with the following hyper-parameters: λ = 0.1, γ = 0.9 and σ = 0.3.

The performance of model uncertainty is evaluated for all the abovementioned GNN models using the same evaluation metrics as preceded as shown in Figs. 7, 8, 9, 10, 11, and 12.

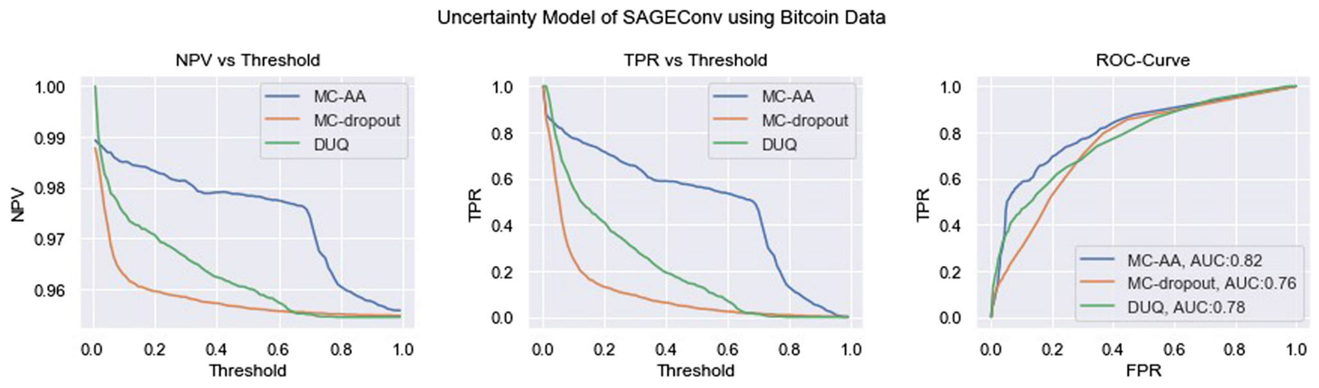

Fig. 4 Model uncertainty of SAGEConv using the Elliptic dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

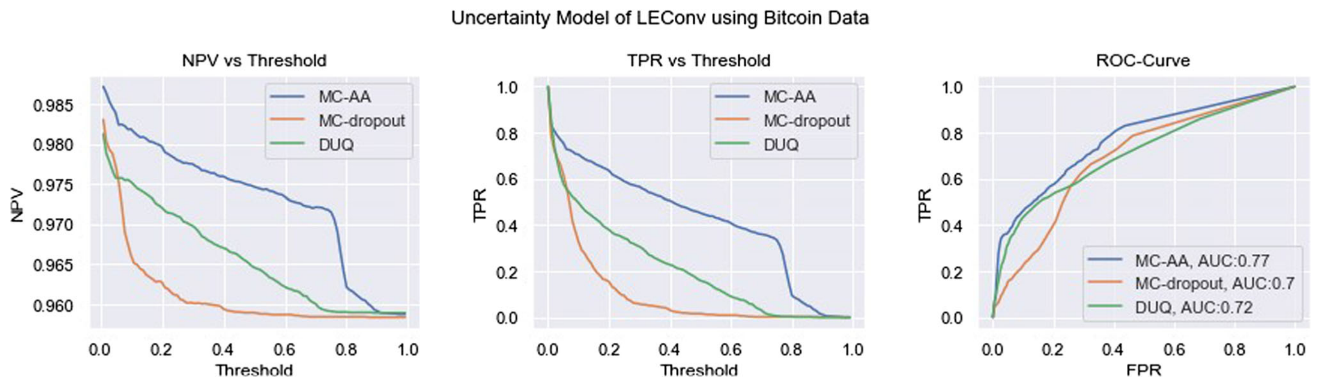

Fig. 5 Model uncertainty of LEConv using the Elliptic dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

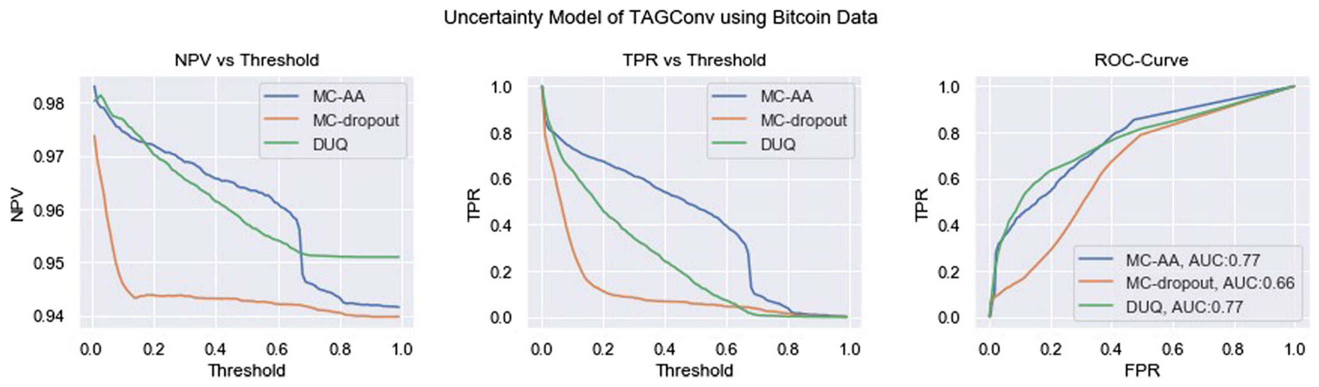

Fig. 6 Model uncertainty of TAGConv using the Elliptic dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

6 Discussion

After conducting comprehensive experiments using several GNN models on Elliptic and GitHub graph data, MC-AA has generally revealed superior success over MC-dropout and DUQ uncertainty methods. From Figs. 1, 2, 3, 4, 5, 6, and 7, NPV and TPR curves are significantly improved with the MC-AA method in all graph learning models using the Elliptic dataset. This shows that MC-AA has detected more data points that are correct knowing that they are certain. Also, higher uncertain data points which are incorrect are detected with MC-AA in comparison to other methods. As a result, the AUC scores, corresponding to the MC-AA method, have revealed a significant outperformance against MC-dropout and DUQ methods.

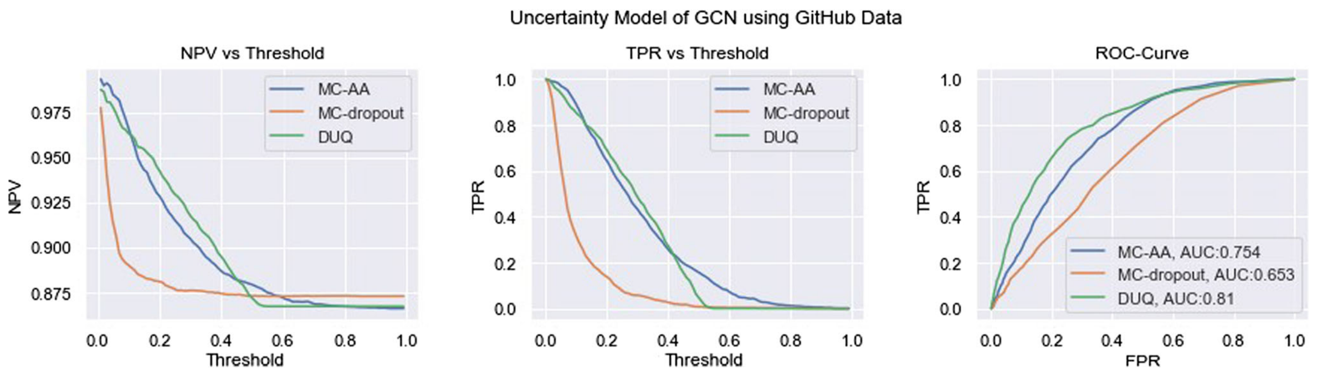

Fig. 7 Model uncertainty of GCN using the GitHub dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

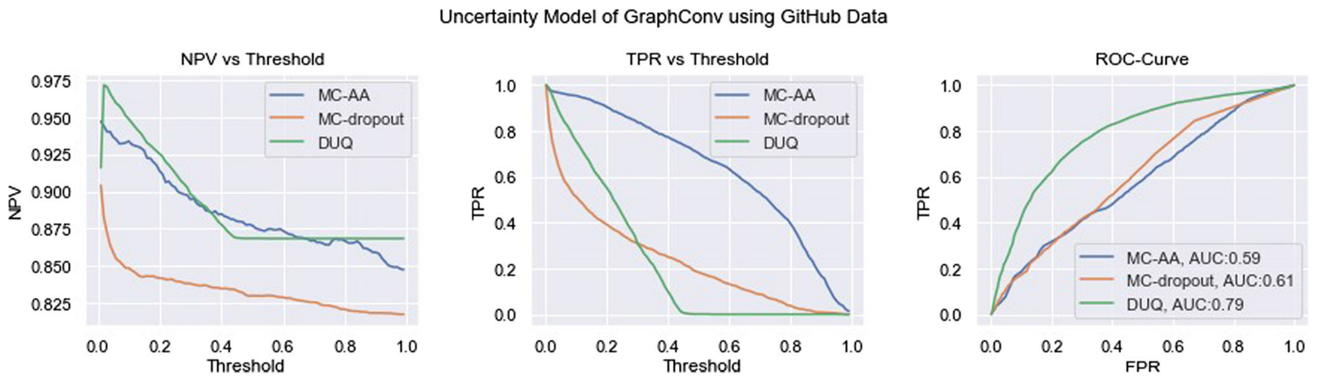

Fig. 8 Model uncertainty of GraphConv using the GitHub dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

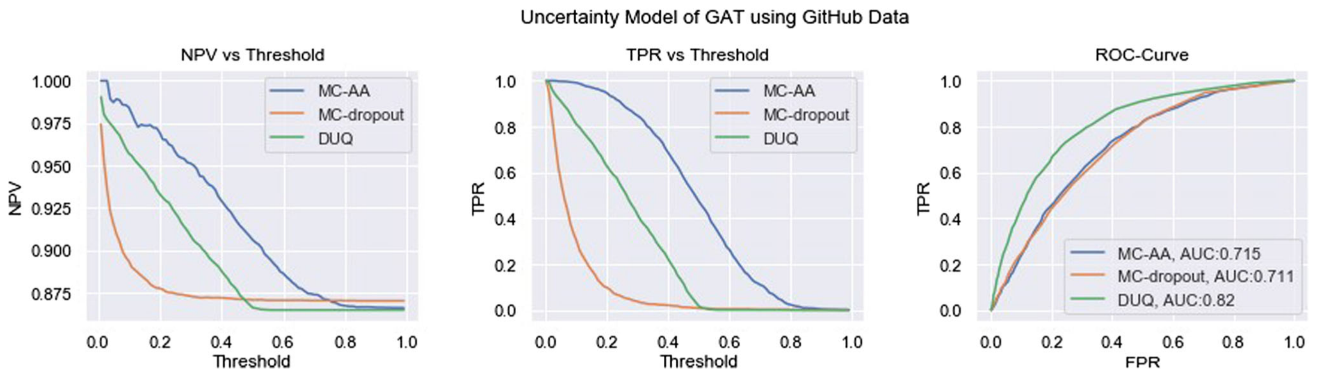

Fig. 9 Model uncertainty of GAT using the GitHub dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

With the GitHub dataset, NPV and TPR have generally recorded better results with MC-AA against other uncertainty methods. However, DUQ has revealed higher AUC scores compared to other uncertainty estimation methods. This means that the number of FN instances (incorrect and certain) is lower with MC-AA but with more FP instances (correct and uncertain). Meanwhile, FP instances (correct and uncertain) do not affect model uncertainty as it is acceptable to have correct predictions that the model is uncertain about. Since MC-AA applies direct perturbations on the data points, the points lying near the decision boundary or in overlapping classes are included as uncertain. Thus, more points around the classifier are subjected to uncertain predictions that have been deduced by the reduced FN and increased FP. Consequently, the behaviour of MC-AA is viable and effective on GNN models that are tested with two datasets, Elliptic and GitHub, from different fields of blockchain and social networks,

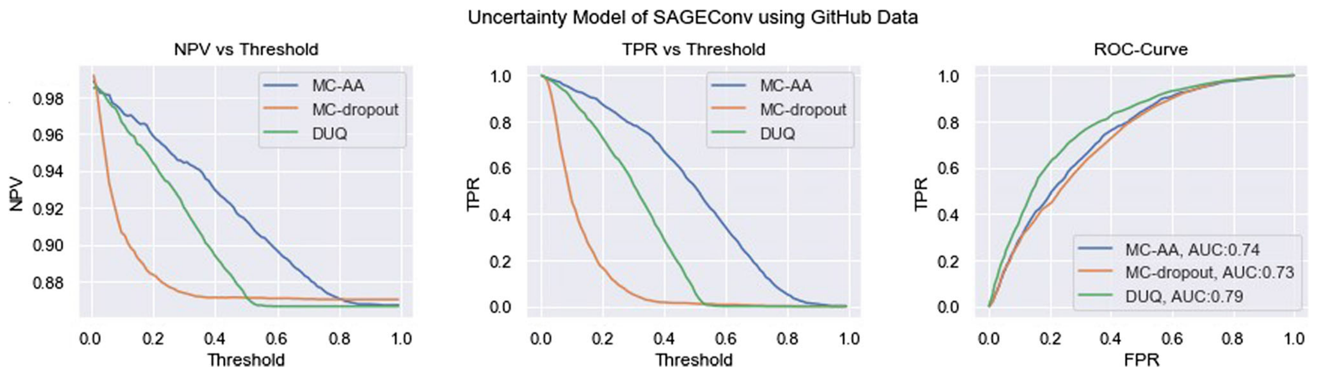

Fig. 10 Model uncertainty of SAGEConv using the GitHub dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

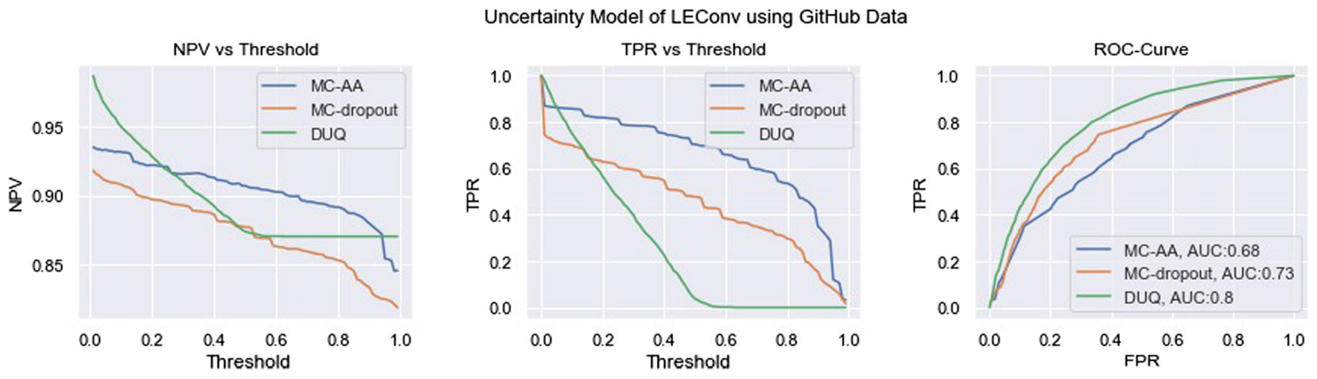

Fig. 11 Model uncertainty of LEConv using the GitHub dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

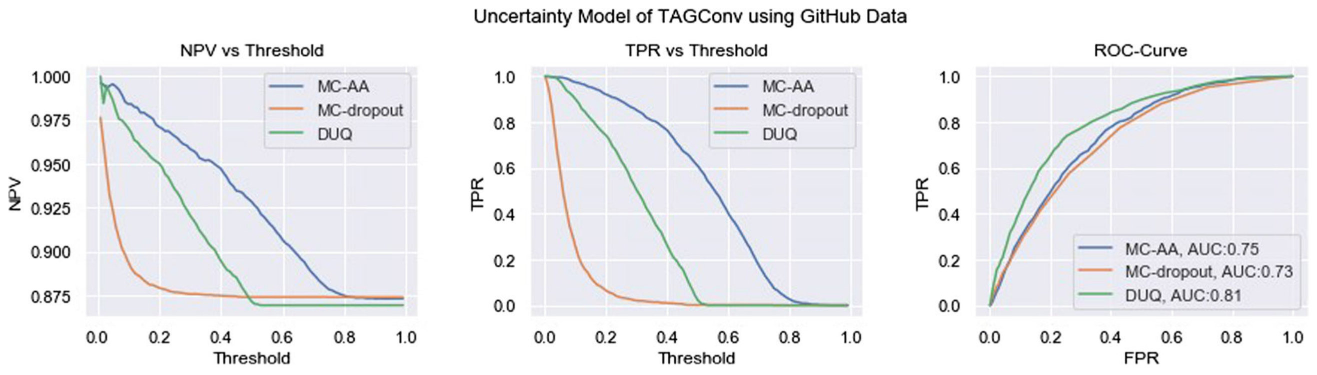

Fig. 12 Model uncertainty of TAGConv using the GitHub dataset. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u

respectively. Generally, MC-AA has attained very reasonable and consistent results in capturing model uncertainty of GNN models. The concept of the MC-AA method is a valid and viable approach which is not limited to given models or graph datasets used in this paper. However, this paper targets binary node classification tasks only. In this paper, both the MC-AA and the MC-dropout method are set with an equal number of forward passes, in which the time complexity for both algorithms is the same. Whereas the DUQ model provides uncertainty with a single forward pass only which is more computationally efficient than the latter algorithms.

We also highlight the node classification results, referring to Tables 4 and 5, in which DeepGCN and TAGConv have performed the best in binary node classification of Elliptic and GitHub datasets with accuracies of 97.1% and 87.4% and f_1-scores of 74.9% and 72.9%, respectively.

7 Conclusion

We have examined the viability of the MC-AA method to capture model uncertainty in graph neural networks (GNNs). We have carried out a comprehensive study on several graph-based approaches and two real-world graph data known as Elliptic and GitHub. Subsequently, we performed uncertainty estimation using MC-AA against MC-dropout and DUQ methods. The evaluation of model uncertainty has generally revealed a significant outperformance of MC-AA over other methods in all GNN models. MC-AA has shown great impact in targeting the erroneous data points that fall between the overlapping classes. As a result, we have concluded that MC-AA is a viable and effective method to capture model uncertainty in GNN models in binary node classification that is not limited to the preceding datasets. We foresee future work to extend our study of MC-AA with multiclass node classification. Furthermore, we plan to include active learning in graph neural networks using the uncertainty measurements of the MC-AA method.

Author's contribution IA: Provided the conception and design of the study, acquisition of data, analysis, and interpretation of data, drafting the article, revised it critically for important intellectual content, and final approval of the version to be submitted; SP: Provided supervision in the concept and design of the study, revised it critically for important intellectual content and gave final approval of the version to be submitted.

Funding No funding to declare.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

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

Ethical approval This paper does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

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

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

Alarab I, Prakoonwit S (2021) Adversarial attack for uncertainty estimation: Identifying critical regions in neural networks. arXiv:2107.07618

Alarab I, Prakoonwit S, Nacer MI (2020a) Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain, In: Proceedings of the 2020a 5th international conference on machine learning technologies, pp 23–27

Alarab I, Prakoonwit S, Nacer MI (2020b) Comparative analysis using super- vised learning methods for anti-money laundering in bitcoin, In: Proceedings of the 2020b 5th international conference on machine learning technologies, pp 11–17

Chakraborty A, Alam M, Dey V, Chattopadhyay A, Mukhopadhyay D (2018) Adversarial attacks and defences: a survey, arXiv preprint arXiv:1810.00069

Du J, Zhang S, Wu G, Moura JMF, Kar S (2018) Topology adaptive graph convolutional networks. arXiv:1710.10370

Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric, In: ICLR workshop on representation learning on graphs and manifolds

Gal Y (2016) Uncertainty in deep learning. Univ Camb 1(3):4

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

Gal Y, Hron J, Kendall A (2017) Concrete dropout, arXiv preprint arXiv:1705.07832

Github social network. URL https://snap.stanford.edu/data/github-social.html

Guo Z, Wang H (2020) A deep graph neural network-based mechanism for social recommendations. IEEE Trans Ind Inf 17(4):2776–2783

Hamilton WL, Ying R, Leskovec J (2018) Inductive representation learning on large graphs. arXiv:1706.02216

Harada S, Akita H, Tsubaki M, Baba Y, Takigawa I, Yamanishi Y, Kashima H (2020) Dual graph convolutional neural network for predicting chemical networks. BMC Bioinform 21(3):1–13

Kennamer N, Ihler AT, Kirkby D (2019) Empirical study of mc-dropout in various astronomical observing conditions. In: CVPR workshops, pp 17–20

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolu- tional networks. arXiv:1609.02907

Lim J, Ryu S, Park K, Choe YJ, Ham J, Kim WY (2019) Predicting drug—target interaction using a novel graph neural network with 3d structure embedded graph representation. J Chem Inf Model 59(9):3981–3988

MacKay DJ (1992) A practical bayesian framework for backpropagation net-works. Neural Comput 4(3):448–472

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

Morris C, Ritzert M, Fey M, Hamilton WL, Lenssen JE, Rattan G, Grohe M (2020) Weisfeiler and leman go neural: higher-order graph neural networks. arXiv:1810.02244

Neal RM (2012) Bayesian learning for neural networks, vol 118. Springer Sci-ence & Business Media, Berlin

Ng M, Guo F, Biswas L, Wright GA (2018) Estimating uncertainty in neu- ralnetworks for segmentation quality control, Tech. rep., Technical report

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'Alché-Buc F, Fox E, Garnett R (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Ranjan E, Sanyal S, Talukdar PP (2020) Asap: adaptive structure aware pooling for learning hierarchical graph representations. arXiv:1911.07979

Rasmussen CE (2003) Gaussian processes in machine learning, In: Summer school on machine learning, Springer, pp 63–71

Rozemberczki B, Allen C, Sarkar R (2019) Multi-scale attributed node embed- ding. arXiv:1909.13021

Smith L, Gal Y (2018) Understanding measures of uncertainty for adversarial example detection, arXiv preprint arXiv:1803.08533

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

van Amersfoort J, Smith L, Jesson A, Key O, Gal Y (2021) Improving de- terministic uncertainty estimation in deep learning for classification and regression, arXiv preprint arXiv:2102.11409

Vlickovic P, Cucurull G, Casanova A, Romero A, Li'o P, Bengio Y (2018) Graph attention networks. arXiv:1710.10903

Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robin-son T, Leiserson CE (2019) Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics, arXiv preprint arXiv:1908.02591

Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks?, arXiv preprint arXiv:1810.00826