

# On Calibration of Graph Neural Networks for Node Classification

1<sup>st</sup> Tong Liu  
LMU Munich  
Munich, Germany  
tong.liu@physik.uni-muenchen.de

1<sup>st</sup> Yushan Liu  
Siemens AG, LMU Munich  
Munich, Germany  
yushan.liu@siemens.com

2<sup>nd</sup> Marcel Hildebrandt  
Siemens AG  
Munich, Germany  
marcel.hildebrandt@siemens.com

3<sup>rd</sup> Mitchell Joblin  
Siemens AG  
Munich, Germany  
mitchell.joblin@siemens.com

4<sup>th</sup> Hang Li  
Siemens AG, LMU Munich  
Munich, Germany  
hang.li@siemens.com

5<sup>th</sup> Volker Tresp  
Siemens AG, LMU Munich  
Munich, Germany  
volker.tresp@siemens.com

**Abstract**—Graphs can model real-world, complex systems by representing entities and their interactions in terms of nodes and edges. To better exploit the graph structure, graph neural networks have been developed, which learn entity and edge embeddings for tasks such as node classification and link prediction. These models achieve good performance with respect to accuracy, but the confidence scores associated with the predictions might not be calibrated. That means that the scores might not reflect the ground-truth probabilities of the predicted events, which would be especially important for safety-critical applications. Even though graph neural networks are used for a wide range of tasks, the calibration thereof has not been sufficiently explored yet. We investigate the calibration of graph neural networks for node classification, study the effect of existing post-processing calibration methods, and analyze the influence of model capacity, graph density, and a new loss function on calibration. Further, we propose a topology-aware calibration method that takes the neighboring nodes into account and yields improved calibration compared to baseline methods.

**Index Terms**—Graph neural networks, calibration, node classification

## I. INTRODUCTION

Learning graph representations for relational data structures has been gaining increasing attention in the machine learning community [1], [2]. A graph is able to model real-world, complex systems by representing entities as nodes and interactions between them as edges. Since the information in graphs is often incomplete, e.g., missing node attributes or edges, relevant graph-related tasks for attaining new knowledge include node classification and link prediction. A variety of graph neural network (GNN) models have been developed [3]–[5], which learn node and edge embeddings in a low-dimensional vector space. Subsequently, these embeddings can be used to solve downstream tasks like node classification. Usually, the focus here lies on maximizing the accuracy – the proportion of nodes that are classified correctly. GNNs achieve good performance with respect to accuracy but are also black boxes and lack interpretability.

Most machine learning models output confidence scores associated with the predictions, and the concept of calibration

captures the idea that the score should reflect the ground-truth probability of the prediction’s correctness. For example, if 100 instances have a score of 0.6 for a specific class  $k$ , then 60 instances are expected to actually be of class  $k$ . A real-world application is autonomous driving, where the model should not only be aware that the object in front of the car is more likely to be a plastic bag than a pedestrian but also know how much more likely it is. A score distribution of 0.99 for plastic bag and 0.01 for pedestrian or 0.51 for plastic bag and 0.49 for pedestrian could have a huge influence on the next action of the car. Generally, calibrated scores lead to a better interpretation of the results and increase the trustworthiness of machine learning models, which is especially important in safety-critical domains.

The calibration of deep neural networks has been addressed in several works [6]–[9]. The calibration of GNNs, however, has not been sufficiently explored yet, and existing calibration methods do not exploit the graph structure. Due to the different architectures of GNNs compared to neural networks, GNNs might exhibit different calibration characteristics. In this work, we are interested in the following research questions:

**R1.** How are GNNs calibrated for the node classification task, and are existing calibration methods sufficient to calibrate GNNs?

**R2.** How do model capacity (width and depth) and graph density influence the calibration?

**R3.** Can a calibration error term be added to the loss function in a straightforward way to improve the calibration without hurting the accuracy?

**R4.** Can the incorporation of topological information improve calibration?

To better understand the calibration properties of GNNs, we conduct an empirical analysis of several GNN models in a node classification setting. Based on our experimental finding that the nodes in the graph express different levels of over- and underconfidence, we propose a topology-aware calibration method that takes the neighboring nodes into account. Our contributions are summarized as follows:

- We inspect the calibration of five representative GNN models on three benchmark citation datasets for node classification.
- We analyze the influence of model capacity, graph density, and a new loss function on the calibration of GNNs.
- We propose a calibration method that takes the graph topology into account and yields improved calibration compared to state-of-the-art post-processing calibration methods.

In Section II, we define the necessary concepts and summarize related work. The existing GNNs and calibration methods used in this work are also described briefly. An experimental study on the calibration of GNNs is presented in Section III ( $\rightarrow$  **R1**, **R2**, **R3**). In Section IV ( $\rightarrow$  **R1**, **R4**), we propose a topology-aware calibration method and show experimental results compared to state-of-the-art calibration baselines. The results are discussed in Section V.

## II. BACKGROUND

### A. Definitions

1) *Node classification on graphs*: An undirected graph is defined as  $G = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  is the set of nodes and  $\mathcal{E}$  the set of edges. An edge  $e = \{i, j\} \in \mathcal{E}$  connects the two nodes  $i$  and  $j$  in the graph. The information about the edges can be encoded in an adjacency matrix  $\mathbf{A} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ . With  $\mathbf{A}_{ij}$  being the entry in the  $i$ -th row and  $j$ -th column of  $\mathbf{A}$ , we define  $\mathbf{A}_{ij} = 1$  if  $\{i, j\} \in \mathcal{E}$  and  $\mathbf{A}_{ij} = 0$  otherwise<sup>1</sup>. Moreover, we define  $\mathcal{N}(i)$  as the set of neighbors of node  $i$ . For attributed graphs, where each node  $i$  is associated with a  $d$ -dimensional feature vector  $\mathbf{X}_i \in \mathbb{R}^d$ , we denote the feature matrix by  $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ .

The goal of the node classification task is to assign each node  $i \in \mathcal{V}$  a class label  $\hat{y}_i \in \mathcal{K} := \{1, 2, \dots, K\}$ , where  $K$  stands for the total number of classes.

2) *Calibration*: Let  $\mathbf{H}_i \in \mathbb{R}^h$  denote the node embedding and  $y_i \in \mathcal{K}$  the ground-truth label of sample (or node)  $i \in \mathcal{V}$ . Let  $g: \mathbb{R}^h \rightarrow [0, 1]^K$  be a function that takes  $\mathbf{H}_i$  as input and outputs a probability vector  $g(\mathbf{H}_i)$ , where  $g(\mathbf{H}_i)_k$  represents the  $k$ -th element. The predicted class label for sample  $i$  is given by  $\hat{y}_i = \arg \max_{k \in \mathcal{K}} g(\mathbf{H}_i)_k$ , where  $\hat{p}_i = \max_{k \in \mathcal{K}} g(\mathbf{H}_i)_k$  is called the corresponding confidence score for  $\hat{y}_i$ . Perfect calibration is defined as  $\mathbb{P}(\hat{y}_i = y_i \mid \hat{p}_i = p) = p$  for all  $p \in [0, 1]$  and any sample  $i$  [6].

A reliability diagram [10] plots accuracy against confidence to visualize the calibration of the model (see Fig. 1). More formally, the samples are grouped into  $M \in \mathbb{N}$  equally-spaced interval bins according to their confidences  $\hat{p}_i$ . For each bin  $B_m$ ,  $m \in \{1, 2, \dots, M\}$ , the accuracy and average confidence are calculated according to

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}[\hat{y}_i = y_i] \quad \text{and} \quad (1)$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \quad (2)$$

<sup>1</sup>We identify nodes and indices to ease the notation.

respectively, where  $|B_m|$  denotes the number of samples in bin  $B_m$  and  $\mathbb{1}$  the indicator function. In case of perfect calibration, the equation  $\text{acc}(B_m) = \text{conf}(B_m)$  holds for all  $m$ . Reliability diagrams also present a way to identify if the model is over- or underconfident. If the bars are above the diagonal line, it implies that the accuracy is higher than the average confidence, and the model is called underconfident. If the bars are below the diagonal, the model is overconfident.

The expected calibration error (ECE) [11] measures the miscalibration by averaging the gaps in the reliability diagram and is given by

$$\sum_{m=1}^M \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|, \quad (3)$$

where  $N$  is the total number of samples.

The marginal ECE (MECE) approximates the marginal calibration error [12], which takes all classes into account. For each bin and every class  $k$ , it compares the average confidence of samples for class  $k$  to the proportion of samples that has as ground-truth label class  $k$ . The MECE is defined as

$$\sum_{k=1}^K w_k \sum_{m=1}^M \frac{1}{N} \left| \sum_{i \in B_m} \mathbb{1}[y_i = k] - \sum_{i \in B_m} g(\mathbf{H}_i)_k \right|, \quad (4)$$

where  $w_k$  is a class-dependent weight factor, which is set to  $1/K$  if all classes are equally important.

### B. Related work

Guo et al. [6] showed that modern neural networks are miscalibrated and tend to be overconfident, i. e., the confidence scores are higher than the proportions of correct predictions. They proposed temperature scaling, a single-parameter variant of Platt scaling [13], to calibrate the results. Several other methods were introduced to improve the calibration of deep neural networks (e. g., mixup training [7] and FALCON [8]). Methods that improve calibration by preventing overconfidence include label smoothing [9] and focal loss [14], [15]. In GNNs, calibration issues have only been studied recently. A first evaluation of GNNs was done by Teixeira et al. [16], who performed experiments on multiple node classification datasets and concluded that GNNs are miscalibrated and existing calibration methods are not always able to improve the calibration to the desired extent.

### C. Methods

1) *Graph neural networks*: Given an adjacency matrix  $\mathbf{A}$  and a feature matrix  $\mathbf{X}$ , the idea of all GNNs is to learn node embeddings  $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times h}$ . The embedding for node  $i$  is denoted by  $\mathbf{H}_i \in \mathbb{R}^h$ , which can be fed to a task-specific decoder  $g$ . For example, since we are concerned with node classification, we use a single-layer perceptron with softmax activation as decoder. For our experiments, we select the widely used models graph convolutional network (GCN) [3], graph attention network (GAT) [4], and simple graph convolution (SGC) [17]. Further, we consider graph filter neural network (gfNN) [18], a straightforward extension of SGC, and

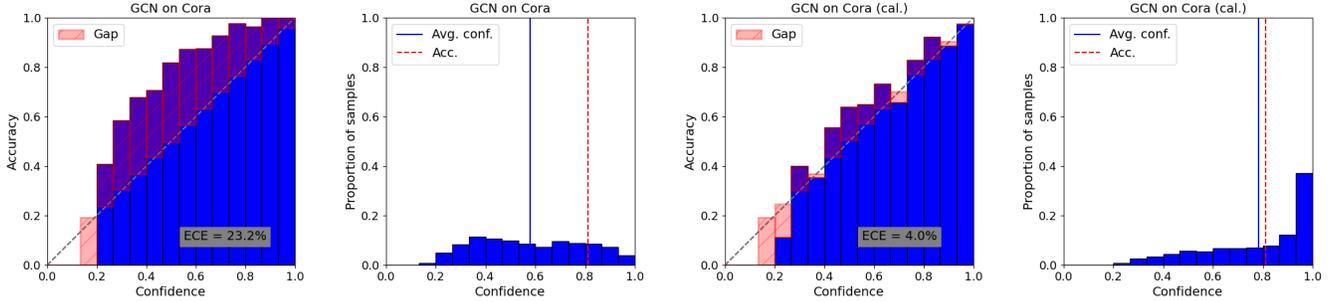


Fig. 1. Reliability diagrams and corresponding confidence histograms for GCN on Cora. The two left plots show the results before calibration, while the two right plots show the results after calibration with temperature scaling. The diagonal line indicates perfect calibration.

approximate personalized propagation of neural predictions (APPNP) [5], a model with state-of-the-art performance.

*GCN* applies a normalized adjacency matrix with self-loops  $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\hat{\mathbf{D}}^{-\frac{1}{2}}$ , where  $\mathbf{I}$  is the identity matrix and  $\hat{\mathbf{D}}$  the degree matrix of  $\mathbf{A} + \mathbf{I}$ . Concretely, the hidden layer of a GCN is formed according to  $\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$ , where  $\mathbf{W}^{(l)}$  is a trainable weight matrix,  $\sigma$  an activation function, and  $\mathbf{H}^{(0)} := \mathbf{X}$ . GCN aggregates information from a node’s neighbors by computing the normalized sum of adjacent node embeddings.

*GAT* differs from GCN in the neighbor aggregation function by introducing an attention mechanism that scales the importance of neighbors when summing over their embeddings.

*SGC* is a GCN without nonlinear activation functions between the layers, resulting from the authors’ conjecture that the good performance of GCNs comes from the aggregation of local neighborhood information and not from the application of nonlinear feature maps.

*gfNN* extends SGC with a nonlinear layer  $\sigma$  so that the node embeddings for layer  $l$  are obtained from  $\mathbf{H}^{(l)} = \sigma(\tilde{\mathbf{A}}^l \mathbf{X} \mathbf{W})$ , where  $\mathbf{W}$  is a trainable weight matrix.

*APPNP* is based on the personalized PageRank (PPR) algorithm [19]. The node embeddings in layer  $l + 1$  are calculated via  $\mathbf{H}^{(l+1)} = (1 - \alpha)\tilde{\mathbf{A}}\mathbf{H}^{(l)} + \alpha\mathbf{H}^{(0)}$ , where  $\alpha \in (0, 1]$  is a hyperparameter and  $\mathbf{H}^{(0)} := f(\mathbf{X})$ , with  $f$  being a trainable neural network.

2) *Calibration methods*: We consider the classical post-processing methods histogram binning [20], isotonic regression [21], and Bayesian binning into quantiles (BBQ) [11], which is a refinement of histogram binning. Further, we include temperature scaling [6] as a multiclass calibration method and Meta-Cal [22], a recently introduced approach with state-of-the-art performance.

*Histogram binning* divides the confidence scores  $\hat{p}_i$  into  $M$  bins and assigns a new score  $\hat{q}_m$  to each bin to represent the calibrated confidences. The scores  $\hat{q}_m$  are learned by minimizing  $\sum_{m=1}^M \sum_{i \in B_m} (\hat{q}_m - y_i)^2$ .

*Isotonic regression* learns a piecewise constant function  $f$  by minimizing  $\sum_{m=1}^M \sum_{i \in B_m} (f(\hat{p}_i) - y_i)^2$ . It is a generalization of histogram binning where the bin boundaries and scores are jointly optimized.

*BBQ* extends histogram binning and learns a distribution  $\mathbb{P}(\hat{q}_i | \hat{p}_i, \mathcal{D}_{\text{val}})$  by marginalizing out all possible binnings, where  $\mathcal{D}_{\text{val}}$  is the validation set.

*Temperature scaling* is a single-parameter extension of Platt scaling [13] for multiple classes. Given the output logit vector  $\mathbf{z}$  before the softmax activation, a rescaling  $\mathbf{z}/T$  depending on a temperature  $T > 0$  is applied.

*Meta-Cal* combines temperature scaling as a base model with a bipartite ranking model to weaken the limitation of accuracy-preserving calibration methods. By investigating two practical constraints (miscoverage rate control and coverage accuracy control), the goal is to improve calibration depending on the bipartite ranking while controlling the accuracy.

### III. EXPERIMENTAL STUDY

#### A. Setup

**Experiments** We first inspect the calibration of GNN models on benchmark citation datasets, where we take the best hyperparameter and training settings from the corresponding original papers.

Then, we empirically analyze the influence of model capacity (width and depth) on calibration. It has been observed that stacking too many GCN layers drastically worsens the performance, which is partly attributed to a phenomenon called oversmoothing [23]. Oversmoothing happens when repeated neighbor aggregation leads to similar node embeddings in the graph, and various methods have been proposed to tackle this problem [5], [24]. In the following, we investigate if increasing model depth also affects calibration.

One of the core mechanisms of GNNs is the message aggregation from neighboring nodes. We examine how graph density, i.e., the ratio of the number of edges in the graph to the number of maximum possible edges, influences the calibration performance.

Finally, we also test a new loss function (5) that combines the standard cross-entropy loss  $L_{\text{ce}}$  with an ECE-inspired term  $L_{\text{cal}}$  for optimizing the calibration. We define  $L_{\text{cal}}$  as the cross entropy between the confidence of the sample and the accuracy of its corresponding bin, where the idea is that the confidence should stay close to the accuracy. Given the original cross-entropy loss  $L_{\text{ce}}$ , we define the new loss as

TABLE I  
DATASET STATISTICS.

Dataset	$K$	$d$	$ \mathcal{V} $	$ \mathcal{E} $	Label rate
Cora	7	1,433	2,708	5,429	0.052
Citeseer	6	3,703	3,327	4,732	0.036
Pubmed	3	500	19,717	44,338	0.003

$$L = \alpha L_{ce} + (1 - \alpha)L_{cal} \quad \text{with} \quad (5)$$

$$L_{cal} = - \sum_{i=1}^N \text{acc}(B_m(i)) \cdot \log(\hat{p}_i), \quad (6)$$

where  $\alpha \in (0, 1)$  and  $B_m(i)$  denotes the bin that sample  $i$  belongs to.

For the experiments on width, depth, graph density, and the new loss function, we focus on GCN and GAT, two of the basic and most widely used GNN models.

**Datasets** Cora, Citeseer, and Pubmed<sup>2</sup> are three commonly used benchmark datasets for node classification. They are citation networks, where nodes represent scientific publications and edges between pairs of nodes correspond to one publication citing the other. Each node comes with a  $d$ -dimensional feature vector that indicates the presence of words from a predefined vocabulary. The class label of a node is the topic of the corresponding publication. Similar to previous works [3], [4], we operate under a semi-supervised setting, where only a small amount of labeled data is available during training. The statistics of the datasets are summarized in Table I.

**Implementation** The GNN models are implemented using the PyTorch-Geometric library<sup>3</sup>. The bin number for calculating the ECE and MECE is set to 15. More information about the hyperparameters and experimental settings can be found in the supplementary material<sup>4</sup>.

## B. Results

**Uncalibrated results** We run all GNNs on the three citation datasets and show the uncalibrated performance with respect to accuracy, ECE, and MECE in Table II. The method APPNP is best on Cora and Pubmed in terms of accuracy (second-best on Citeseer), and it is also best calibrated on the datasets Citeseer and Pubmed. For Cora, gfNN has the lowest ECE and MECE. All models except for gfNN<sup>5</sup> have stable calibration values with small standard deviations. The worst method with respect to the calibration performance is SGC, which is, apart from the softmax activation for normalization, the only linear model. Adding a nonlinear layer as in gfNN results in better calibration. Moreover, we find that GAT outperforms GCN in terms of ECE and MECE in two of three datasets.

**Influence of width** We compare the calibration of GCN and GAT for varying model width, i.e., the number of hidden

<sup>2</sup><https://github.com/kimiyoung/planetoid>

<sup>3</sup>[https://github.com/pyg-team/pytorch\\_geometric](https://github.com/pyg-team/pytorch_geometric)

<sup>4</sup>Source code and supplementary material available at <https://github.com/liu-yushan/calGNN>.

<sup>5</sup>The original paper trains for 50 epochs without early stopping. A different training setting might stabilize the results more.

TABLE II  
UNCALIBRATED PERFORMANCE WITH RESPECT TO ACCURACY, ECE, AND MECE (MEAN $\pm$ SD OVER 100 INDEPENDENT RUNS). THE BEST RESULTS ARE DISPLAYED IN BOLD.

Dataset	Model	Acc.	ECE	MECE
Cora	GCN	81.43 $\pm$ 0.60	23.51 $\pm$ 1.89	7.01 $\pm$ 0.46
	GAT	83.14 $\pm$ 0.39	17.26 $\pm$ 1.09	5.15 $\pm$ 0.30
	SGC	81.19 $\pm$ 0.05	26.03 $\pm$ 0.16	7.78 $\pm$ 0.08
	gfNN	78.73 $\pm$ 5.04	<b>6.45<math>\pm</math>2.44</b>	<b>3.16<math>\pm</math>1.45</b>
	APPNP	<b>83.68<math>\pm</math>0.36</b>	14.90 $\pm$ 0.69	4.73 $\pm$ 0.17
Citeseer	GCN	71.32 $\pm$ 0.70	21.80 $\pm$ 1.21	8.57 $\pm$ 0.32
	GAT	70.99 $\pm$ 0.60	18.92 $\pm$ 1.05	7.66 $\pm$ 0.30
	SGC	<b>72.46<math>\pm</math>0.15</b>	53.59 $\pm$ 0.14	19.15 $\pm$ 0.00
	gfNN	67.33 $\pm$ 6.58	15.50 $\pm$ 4.47	8.40 $\pm$ 1.26
	APPNP	72.10 $\pm$ 0.38	<b>11.93<math>\pm</math>0.80</b>	<b>5.37<math>\pm</math>0.28</b>
Pubmed	GCN	79.23 $\pm$ 0.43	10.62 $\pm$ 1.28	7.29 $\pm$ 0.84
	GAT	79.05 $\pm$ 0.38	14.37 $\pm$ 0.48	9.89 $\pm$ 0.23
	SGC	78.72 $\pm$ 0.04	22.40 $\pm$ 0.04	14.98 $\pm$ 0.02
	gfNN	77.94 $\pm$ 2.32	6.04 $\pm$ 2.90	5.23 $\pm$ 2.65
	APPNP	<b>80.09<math>\pm</math>0.25</b>	<b>4.38<math>\pm</math>0.74</b>	<b>3.59<math>\pm</math>0.41</b>

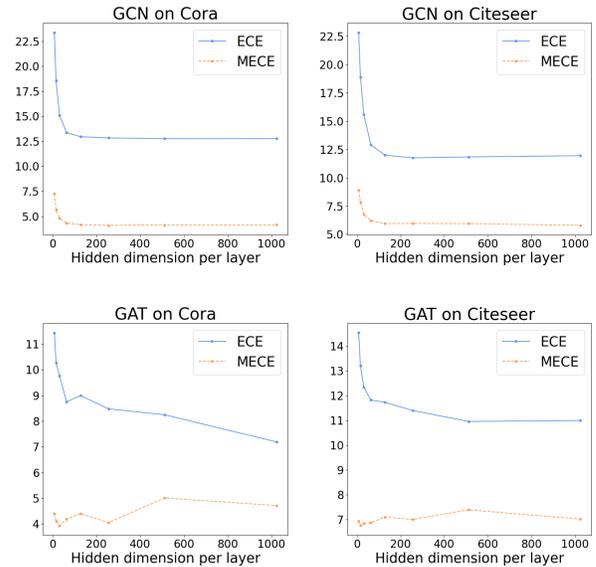


Fig. 2. Varying model width (hidden dimension per layer).

dimensions per layer. While the accuracy basically stays constant, the ECE and MECE decrease with increasing number of hidden dimensions initially (see Fig. 2). When a certain width is reached, the calibration values stagnate or slightly increase again. Generally, wider networks tend to be better calibrated.

**Influence of depth** We investigate the influence of model depth, i.e., the number of layers, on the calibration performance (see Fig. 3). Oversmoothing becomes particularly pronounced when the test accuracy decreases significantly with increasing number of layers. The ECE first improves when changing from two to three layers, then it increases again until five or six layers. Using an even larger model depth, the ECE eventually decreases again.

**Influence of graph density** For this experiment, we remove

TABLE III

UNCALIBRATED PERFORMANCE OF GCN AND GAT UNDER THE STANDARD AND THE NEW LOSS FUNCTION (MEAN±SD OVER 10 INDEPENDENT RUNS). THE BETTER RESULTS WHEN COMPARING THE TWO LOSS FUNCTIONS ARE UNDERLINED.

Dataset	Model	Acc. ( $L_{ce}$ )	ECE ( $L_{ce}$ )	MECE ( $L_{ce}$ )	Acc. ( $L$ )	ECE ( $L$ )	MECE ( $L$ )
Cora	GCN	81.43±0.60	23.51±1.89	7.01±0.46	<u>81.81±0.85</u>	<u>14.91±2.7</u>	<u>4.64±0.56</u>
	GAT	<u>83.14±0.39</u>	17.26±1.09	5.15±0.30	82.73±0.40	<u>5.29±1.31</u>	<u>2.41±0.32</u>
Citeseer	GCN	71.32±0.70	21.80±1.21	8.57±0.32	<u>71.67±0.50</u>	<u>14.65±0.42</u>	<u>6.43±0.26</u>
	GAT	70.99±0.60	18.92±1.05	7.66±0.30	<u>71.13±0.44</u>	<u>9.39±0.97</u>	<u>4.58±0.40</u>
Pubmed	GCN	<u>79.23±0.43</u>	10.62±1.28	7.29±0.84	79.00±0.37	<u>7.50±0.91</u>	<u>5.60±0.73</u>
	GAT	<u>79.05±0.38</u>	14.37±0.48	9.89±0.23	78.88±0.40	<u>9.58±0.79</u>	<u>6.91±0.63</u>

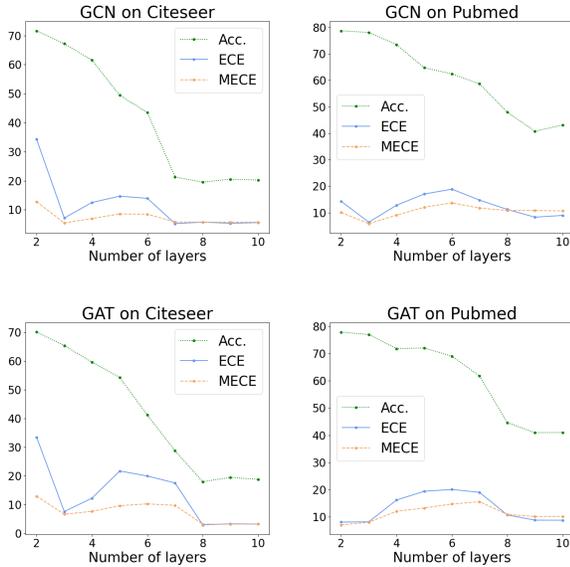


Fig. 3. Varying depth (number of layers).

different proportions of edges randomly from the dataset, ranging from 0% (original dataset) to 100% (no graph structure at all). The models GCN and GAT only differ in the aggregation mechanism, i. e., GAT introduces attention coefficients to weight the importance of neighbors. The results are shown in Fig. 4. Similar to Table II, the ECE of GAT is consistently lower than the ECE of GCN on Cora and Citeseer, while on Pubmed, GCN expresses partly better calibration. Generally, the graph density of Pubmed is the lowest. It might be that the attention weights in GAT are beneficial for calibration and especially useful when enough edges exist in the graph.

**Influence of new loss function** Table III compares the results of the standard cross-entropy loss  $L_{ce}$  and the new loss function  $L$  from (5), which contains a calibration error term. The new loss  $L$  improves the model calibration in all cases while keeping the accuracy at the same level or even slightly increasing the accuracy.

**Underconfidence vs. overconfidence** Taking the best hyperparameter and training settings from their corresponding publications, all GNNs exhibit underconfidence on all three datasets, i. e., the confidence scores are lower than the accuracy of the predictions (see Fig. 1 and figures in the supplementary

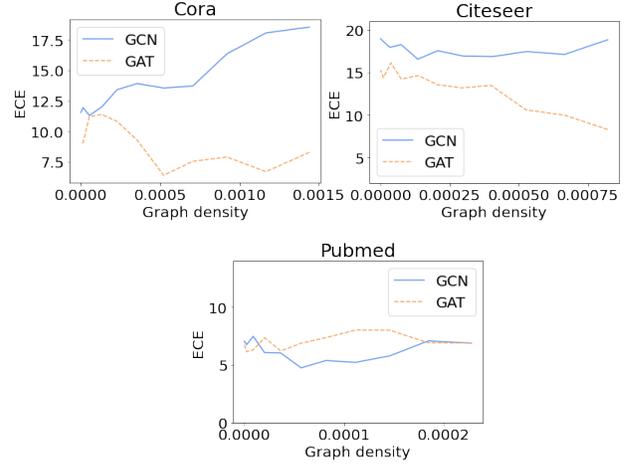


Fig. 4. Influence of graph density. The graph density is the ratio of the number of edges in the graph to the number of maximum possible edges.

material). In some cases, however, we find that the model changes from underconfidence to overconfidence if it is trained without early stopping.

The left plot in Fig. 5 shows the test accuracy, scaled test negative log-likelihood (NLL), and scaled test ECE for a 4-layer GCN on Cora during training, with a weight decay set to  $5e-4$ . Around epoch 150, the NLL and accuracy become stable, while the ECE is still improving. At this point, GCN is underconfident, as shown in the left-most reliability diagram in Fig. 6. In the epochs between 200 and 300, the ECE gains the best performance when the model changes from underconfidence to overconfidence (see the two diagrams in the middle of Fig. 6). After epoch 300, GCN starts to overfit with respect to the ECE, while the NLL and accuracy remain rather unchanged. During this process, overconfidence aggravates, and the ECE increases to 7.8% in epoch 400, which is displayed in the right-most diagram in Fig. 6.

In summary, GCN first optimizes NLL and accuracy during training, then fits the confidence scores, and eventually starts to overfit regarding the ECE without influencing the NLL and accuracy. When we slightly increase the weight decay to  $7.5e-4$  (see right plot in Fig. 5), the ECE stabilizes after reaching the optimal value, and the values of NLL and accuracy also stay in a smaller range compared to the left plot.

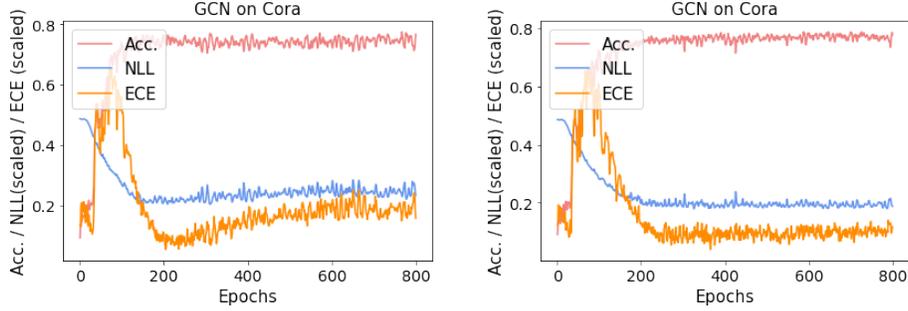


Fig. 5. Test accuracy, scaled test NLL, and scaled test ECE for GCN on Cora, with a weight decay of  $5e-4$  (left) and  $7.5e-4$  (right).

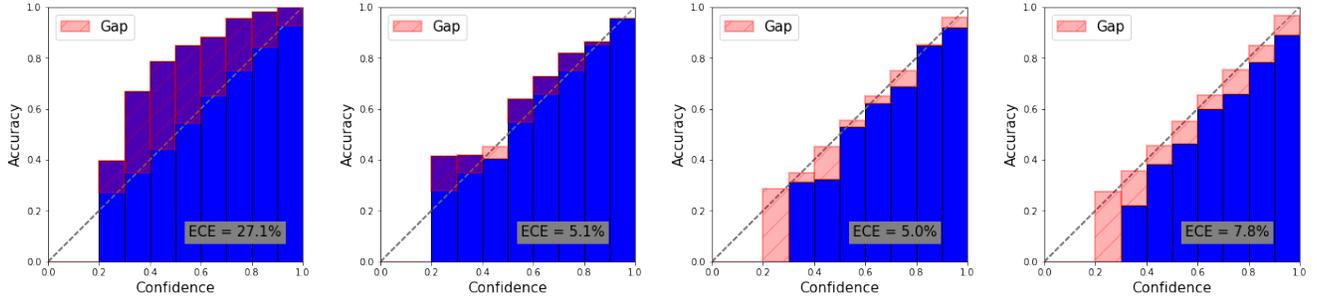


Fig. 6. Reliability diagrams for GCN on Cora during the training process. From left to right, the corresponding number of epochs is 100, 200, 300, and 400.

#### IV. RATIO-BINNED SCALING FOR CALIBRATING GNNs

##### A. Same-class-neighbor ratio

From our experiments, we find that GNN models tend to be underconfident. Even though the overall model exhibits underconfidence, there might be differences depending on node-level properties, which have not been considered before. Especially for graph data, the topology could provide structural information that are useful for calibration. For node classification, the class labels and properties of a node’s neighbors have a significant influence on the classification. We calculate for each node  $i$  the same-class-neighbor ratio, i.e., the proportion of neighbors that have the same class as node  $i$ , and develop a new binning scheme that groups samples into bins based on the same-class-neighbor ratio for calibration.

To evaluate the correlation between the same-class-neighbor ratio and the confidence of a model, we calculate the ratio for each node based on the ground-truth labels. In Fig. 7, we group the nodes into 5 equally-spaced interval bins according to their ratios. Employing a trained GNN model, we compute the output of the classifier  $g$  for each node and draw the average confidence of the samples in each bin as a blue bar. The gap illustrates the difference between the average confidence and the accuracy in each bin. We observe that the average confidence increases with the same-class-neighbor ratio, where bins with higher ratios express underconfidence and bins with lower ratios overconfidence. Consequently, a binning scheme that groups samples depending on their same-class-neighbor ratios would take the graph structure into account and allow

for an adaptive calibration depending on the confidence level of each bin.

##### B. Ratio-binned scaling

We propose ratio-binned scaling (RBS), a topology-aware method, which first approximates the same-class-neighbor ratio for each sample, then groups the samples into  $M$  bins, and finally learns a temperature for each bin to rescale the confidence scores.

In the semi-supervised setting, we only know the labels of a small number of nodes and therefore cannot use the true labels for binning. One natural option is to replace the nodes’ ground-truth labels with their confidence scores for estimating the same-class-neighbor ratio. More precisely, we define the estimated ratio for node  $i$  as

$$\hat{r}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} g(\mathbf{H}_j)_{\hat{y}_i} \in [0, 1], \quad (7)$$

where  $\mathbf{H}_j$  is the node embedding of node  $j$ , which is learned by a GNN model, and  $g$  is the classifier.  $g(\mathbf{H}_j)_{\hat{y}_i}$  denotes the confidence score of node  $j$  corresponding to the class  $\hat{y}_i$  that is predicted for the central node  $i$ .

For the ratio-based binning scheme, let  $\{B_m \mid 1 \leq m \leq M\}$  be a set of bins that partitions the interval  $[0, 1]$  uniformly. After calculating the output for all nodes, each node  $i$  is assigned to a bin according to its estimated same-class-neighbor ratio  $\hat{r}(i)$ , i.e.,  $B_1 = \{i \in \mathcal{V} \mid \hat{r}(i) \in [0, \frac{1}{M}]\}$  and  $B_m = \{i \in \mathcal{V} \mid \hat{r}(i) \in (\frac{m-1}{M}, \frac{m}{M}]\}$  for  $m \in \{2, \dots, M\}$ .

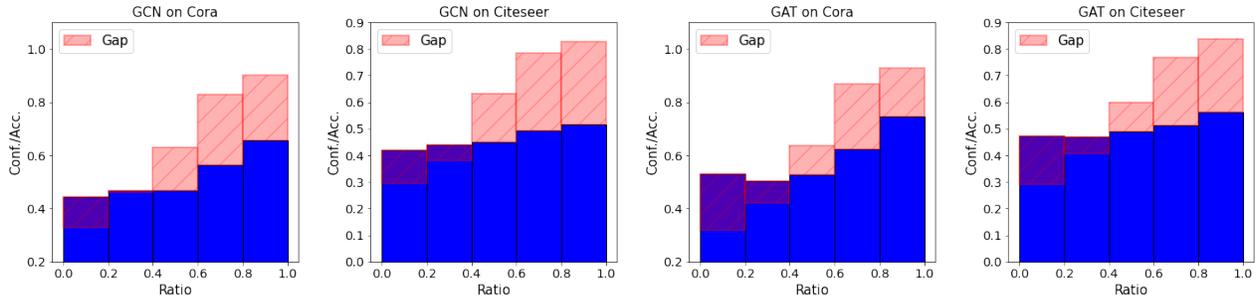


Fig. 7. The nodes are grouped according to their same-class-neighbor ratios. The blue bar represents the average confidence and the gap the difference between average confidence and accuracy in each bin.

TABLE IV  
CALIBRATED PERFORMANCE WITH RESPECT TO ECE (MEAN $\pm$ SD OVER 100 INDEPENDENT RUNS). THE BEST GNN MODEL FOR EACH CALIBRATION METHOD IS UNDERLINED. THE BEST CALIBRATION METHOD FOR EACH GNN MODEL IS DISPLAYED IN BOLD.

Dataset	Model	Uncal.	His. bin.	Iso. reg.	BBQ	Tem. scal.	Meta-Cal	RBS	RRBS
Cora	GCN	23.51 $\pm$ 1.89	4.50 $\pm$ 0.76	3.94 $\pm$ 0.66	4.53 $\pm$ 0.64	<b>3.82<math>\pm</math>0.60</b>	4.09 $\pm$ 0.65	3.90 $\pm$ 0.61	3.10 $\pm$ 0.63
	GAT	17.26 $\pm$ 1.09	4.86 $\pm$ 0.62	4.04 $\pm$ 0.59	4.18 $\pm$ 0.60	3.53 $\pm$ 0.66	<b>3.28<math>\pm</math>0.65</b>	3.34 $\pm$ 0.63	2.67 $\pm$ 0.54
	SGC	26.03 $\pm$ 0.16	4.38 $\pm$ 0.30	4.21 $\pm$ 0.42	4.35 $\pm$ 0.21	4.05 $\pm$ 0.11	<u>4.02<math>\pm</math>0.35</u>	<b>3.55<math>\pm</math>0.07</b>	2.57 $\pm$ 0.11
	gfNN	6.45 $\pm$ 2.44	<u>3.80<math>\pm</math>0.86</u>	<b>3.72<math>\pm</math>0.78</b>	4.15 $\pm$ 1.22	3.74 $\pm$ 1.34	4.07 $\pm$ 1.52	3.77 $\pm$ 0.96	3.39 $\pm$ 1.22
	APPNP	14.90 $\pm$ 0.69	4.20 $\pm$ 0.62	<u>3.43<math>\pm</math>0.60</u>	<u>3.90<math>\pm</math>0.57</u>	<u>3.14<math>\pm</math>0.50</u>	3.48 $\pm$ 0.57	<b>3.01<math>\pm</math>0.53</b>	2.68 $\pm$ 0.44
Citeseer	GCN	21.80 $\pm$ 1.21	<u>4.61<math>\pm</math>0.82</u>	<b>4.46<math>\pm</math>0.93</b>	5.30 $\pm$ 1.09	4.86 $\pm$ 0.76	5.04 $\pm$ 0.90	4.99 $\pm$ 0.75	4.11 $\pm$ 0.88
	GAT	18.92 $\pm$ 1.05	5.00 $\pm$ 0.73	4.90 $\pm$ 0.69	<u>5.04<math>\pm</math>0.71</u>	5.92 $\pm$ 0.58	6.08 $\pm$ 0.61	<b>4.45<math>\pm</math>0.73</b>	4.71 $\pm$ 0.67
	SGC	53.59 $\pm$ 0.14	7.55 $\pm$ 0.23	6.93 $\pm$ 0.19	7.43 $\pm$ 0.13	4.47 $\pm$ 0.19	<u>4.17<math>\pm</math>0.29</u>	<b>4.04<math>\pm</math>0.17</b>	2.97 $\pm$ 0.20
	gfNN	15.50 $\pm$ 4.47	<b>4.74<math>\pm</math>1.00</b>	4.92 $\pm$ 1.03	5.06 $\pm$ 1.37	5.43 $\pm$ 1.27	5.45 $\pm$ 1.32	5.19 $\pm$ 1.31	4.34 $\pm$ 1.27
	APPNP	11.93 $\pm$ 0.80	4.75 $\pm$ 0.85	<b>4.50<math>\pm</math>0.67</b>	5.10 $\pm$ 1.02	4.98 $\pm$ 0.67	5.29 $\pm$ 0.67	5.08 $\pm$ 0.69	3.97 $\pm$ 0.58
Pubmed	GCN	10.62 $\pm$ 1.28	4.69 $\pm$ 0.78	4.76 $\pm$ 0.77	<u>4.69<math>\pm</math>0.74</u>	4.27 $\pm$ 0.61	4.99 $\pm$ 1.11	<b>4.16<math>\pm</math>0.60</b>	3.28 $\pm$ 0.89
	GAT	14.37 $\pm$ 0.48	4.85 $\pm$ 0.89	4.93 $\pm$ 0.78	5.70 $\pm$ 1.03	<u>3.94<math>\pm</math>0.67</u>	4.45 $\pm$ 0.74	<b>3.61<math>\pm</math>0.75</b>	2.56 $\pm$ 0.56
	SGC	22.40 $\pm$ 0.04	<u>4.40<math>\pm</math>0.29</u>	<u>4.29<math>\pm</math>0.21</u>	<u>5.04<math>\pm</math>0.22</u>	4.13 $\pm$ 0.12	4.64 $\pm$ 0.58	<b>4.07<math>\pm</math>0.17</b>	3.01 $\pm$ 0.12
	gfNN	6.04 $\pm$ 2.90	4.98 $\pm$ 1.06	5.00 $\pm$ 0.83	5.15 $\pm$ 1.26	4.97 $\pm$ 1.67	6.06 $\pm$ 1.95	<b>4.91<math>\pm</math>1.65</b>	3.78 $\pm$ 1.03
	APPNP	4.38 $\pm$ 0.74	4.86 $\pm$ 0.75	4.72 $\pm$ 0.57	4.79 $\pm$ 0.84	3.98 $\pm$ 0.59	<u>4.34<math>\pm</math>0.72</u>	<b>3.80<math>\pm</math>0.60</b>	2.60 $\pm$ 0.47

Let the output of  $g$  be in the form  $g(\mathbf{H}_i) = \sigma(\mathbf{Z}_i) \in \mathbb{R}^K$  for node  $i$ , where  $\sigma$  is the softmax function and  $\mathbf{Z}_i$  the logits before normalization. For each bin  $B_m$ ,  $m \in \{1, \dots, M\}$ , a temperature  $T_m > 0$  is learned on the validation dataset.

The calibrated confidence for a test node  $j$  is then given by

$$\hat{\mathbf{q}}_j = \sigma(\mathbf{Z}_j/T_m) \in [0, 1]^K \quad \text{if } j \in B_m. \quad (8)$$

We apply temperature scaling for calibrating the nodes in each bin, but it would also be possible to apply other post-processing calibration methods for obtaining calibrated scores.

### C. Results

Table IV summarizes the calibration performance of all considered post-processing calibration methods and our proposed method RBS in terms of ECE. All methods can improve the calibration of GNN models on Cora and Citeseer. In particular, the obtained ECE for a specific dataset and calibration method is rather similar for all GNNs regardless of the uncalibrated ECE. On Pubmed, most methods have difficulties improving calibration of APPNP, which already has low ECE. RBS gains the best performance in the majority of the cases and outperforms classical temperature scaling in 11 out of 15 experiments.

Next to a good calibration performance, accuracy preservation is desirable for calibration methods. RBS and temperature scaling do not change the ranking of the classes and thus the accuracy stays unchanged. Meta-Cal trades good calibration for lower accuracy, while the other methods yield comparable or even slightly improved accuracy in some cases (see supplementary material).

### D. Effectiveness of real-ratio-binned scaling

Table IV also shows the calibrated results of real-ratio-binned scaling (RRBS), where we assume that the ground-truth labels are available for all nodes. RRBS outperforms the best calibration method in 14 out of 15 experiments. Although the correct labels are not accessible in the semi-supervised setting, the results still indicate the effectiveness of the intuition of our proposed method.

## V. DISCUSSION

In general, the calibration performance depends on the specific GNN model and dataset, where all models perform best on Pubmed (see Table IV). When using the hyperparameter and training settings from the original publications, all GNNs tend to be underconfident on all three datasets, in contrast to the finding that deep neural network models rather exhibit

overconfidence [6], [7]. However, when plotting the reliability diagrams for a varying number of epochs, we observe that in some cases, underconfidence changes to overconfidence when the number of epochs increases. It seems that underconfidence or overconfidence is not necessarily a property of the model architecture but is also dependent on the training setting.

Most GNNs suffer from oversmoothing, which becomes apparent when increasing the number of layers in the model [5], [23], [24]. We observe that for large numbers of layers, the accuracy drops significantly, while the ECE improves. Oversmoothing results in similar node embeddings, which might be uninformative for the model. In this case, the model would most likely learn the distribution of classes in the training data as confidence scores. Therefore, all samples would be grouped into one bin, resulting in low ECE if the test distribution is close to the training distribution. However, such a model does not make use of the underlying graph structure and is probably not useful for application.

The results for RBS and RRBS show the potential of a calibration method that takes the graph structure into account, where the binning scheme is constructed depending on node-level properties. RRBS almost always outperforms RBS, which suggests that RBS might be especially helpful for cases where the estimated ratios are close to the real ratios, i.e., for models with relatively high accuracy. The number of bins for RBS was chosen from  $\{2, 3, 4\}$ , and it seems that even a small number of bins can lead to improved calibration compared to classical temperature scaling. It would further be interesting to apply RBS to other kinds of datasets, e.g., heterophilic graphs, where nodes from different classes are likely to be connected.

## VI. CONCLUSION

We investigated the calibration of graph neural networks for node classification on three benchmark datasets. Graph neural networks seem to be miscalibrated, where the exact calibration depends on both the dataset and the model. Existing post-processing calibration methods are able to alleviate the miscalibration but do not consider the graph structure. Based on our experimental finding that the nodes in the graph express different levels of over- or underconfidence depending on their same-class-neighbor ratios, we proposed the topology-aware calibration method ratio-binned scaling. It takes the predictions of neighboring nodes into account and shows better performance compared to state-of-the-art baselines. For future work, it would be interesting to gain a more theoretical understanding of the calibration properties and conduct experiments on larger and a wider variety of datasets.

## ACKNOWLEDGMENT

This work has been supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of the project RAKI under grant number 01MD19012C.

## REFERENCES

- [1] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *arXiv:2005.03675*, 2021.
- [2] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 40, pp. 52–74, 2017.
- [3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the Fifth International Conference on Learning Representations*, 2017.
- [4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the Sixth International Conference on Learning Representations*, 2018.
- [5] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proceedings of the Seventh International Conference on Learning Representations*, 2019.
- [6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the Thirty-Fourth International Conference on Machine Learning*, 2017.
- [7] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, 2019.
- [8] C. Tomani and F. Buettner, "Towards trustworthy predictions from deep neural networks with fast adversarial calibration," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.
- [9] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" in *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, 2019.
- [10] A. H. Murphy and R. L. Winkler, "Reliability of subjective probability forecasts of precipitation and temperature," *Applied Statistics*, vol. 26, pp. 41–47, 1977.
- [11] M. P. Naeni, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using Bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [12] A. Kumar, P. Liang, and T. Ma, "Verified uncertainty calibration," in *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, 2019.
- [13] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, 1999.
- [14] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. S. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *Proceedings of the Thirty-Fourth Conference on Neural Information Processing Systems*, 2020.
- [15] N. Charoenphakdee, J. Vongkulbhisal, N. Chairatanakul, and M. Sugiyama, "On focal loss for class-posterior probability estimation: A theoretical perspective," in *Proceedings of the 2021 Conference on Computer Vision and Pattern Recognition*, 2021.
- [16] L. Teixeira, B. Jalaian, and B. Ribeiro, "Are graph neural networks miscalibrated?" *arXiv:1905.02296*, 2019.
- [17] F. Wu, T. Zhang, Jr. Souza, A. H. d., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the Thirty-Sixth International Conference on Machine Learning*, 2019.
- [18] H. NT and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv:1905.09550*, 2019.
- [19] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," *Stanford InfoLab*, 1998.
- [20] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
- [21] —, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.
- [22] X. Ma and M. B. Blaschko, "Meta-Cal: Well-controlled post-hoc calibration by ranking," in *Proceedings of the Thirty-Eighth International Conference on Machine Learning*, 2021.
- [23] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI conference on Artificial Intelligence*, 2018.
- [24] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, 2018.

SUPPLEMENTARY MATERIAL

*GNN models* We follow the best hyperparameter and training settings given by the corresponding graph neural network papers, using the PyTorch-Geometric implementation<sup>6</sup> of GCN [3], GAT [4], SGC [17], and APPNP [5]. For SGC, we set the learning rate to 0.2, train for 100 epochs without early stopping, and tune weight decay for 60 iterations using Hyperopt<sup>7</sup>, according to the original paper. We implement gfNN [18] by ourselves and follow the setting in the original paper. In order to capture the variance across different training runs, each model is run for 100 times, and we report the averaged results with standard deviations.

*Width* We conduct experiments on the influence of width on the models GCN and GAT, where we use the best hyperparameter settings and vary the hidden dimensions per layer in the range given by  $\{2^i \mid 3 \leq i \leq 10\}$ . Dropout layers are removed, and the number of epochs is set to 200 with early stopping after 10 epochs without improvement of the validation loss. Each model is run for 10 times.

*Depth* We conduct experiments on the influence of depth on the models GCN and GAT, where we follow the experimental setting in Appendix B by Kipf and Welling [3]. The number of layers is in the range  $\{1, 2, \dots, 10\}$ . Each model is run for 10 times.

*Graph density* We conduct experiments on the influence of graph density on the models GCN and GAT. Different proportions of edges are removed randomly from 0% (original dataset) to 100% (no graph structure at all). Each model is run for 10 times.

*New loss function* We follow the setting by Tomani and Buetner [8], who also introduced an ECE-inspired loss function. An annealing coefficient is specified for the calibration error term since the early epochs are usually used for reaching the cross entropy minimum. More precisely, we define

$$\text{anneal\_coef} = \lambda \cdot \min \left\{ 1, \frac{\text{epoch}}{\text{EPOCHS} \cdot \text{anneal\_max}} \right\}, \quad (9)$$

$$\tilde{L}_{\text{cal}} = \text{anneal\_coef} \cdot L_{\text{cal}}, \quad \text{and} \quad (10)$$

$$L = \alpha \cdot L_{\text{ce}} + (1 - \alpha) \cdot \tilde{L}_{\text{cal}}, \quad (11)$$

where epoch and EPOCHS are the current training epoch and the total number of epochs, respectively, and  $\lambda$ , anneal\_max, and  $\alpha$  are hyperparameters. We set anneal\_max = 1,  $\lambda = 1$ , and tune  $\alpha$  on the validation set in the range  $\{0.95, 0.96, 0.97, 0.98, 0.99\}$ . Each model (fixed hyperparameter setting) is run for 10 times.

<sup>6</sup>[https://github.com/pyg-team/pytorch\\_geometric/tree/master/benchmark/citation](https://github.com/pyg-team/pytorch_geometric/tree/master/benchmark/citation)

<sup>7</sup><https://github.com/hyperopt/hyperopt>

TABLE V  
 CALIBRATED ACCURACY (MEAN $\pm$ SD OVER 100 INDEPENDENT RUNS). TEMPERATURE SCALING AND RBS DO NOT CHANGE THE ACCURACY.

Dataset	Model	Uncal.	His	Iso	BBQ	Meta
Cora	GCN	81.43 $\pm$ 0.60	80.38 $\pm$ 0.82	81.80 $\pm$ 0.57	81.34 $\pm$ 0.67	79.23 $\pm$ 1.61
	GAT	83.14 $\pm$ 0.39	81.39 $\pm$ 0.48	84.05 $\pm$ 0.51	83.52 $\pm$ 0.59	79.99 $\pm$ 1.70
	SGC	81.19 $\pm$ 0.05	79.91 $\pm$ 0.13	81.16 $\pm$ 0.11	79.83 $\pm$ 0.24	78.77 $\pm$ 1.88
	gfNN	78.73 $\pm$ 5.04	79.06 $\pm$ 1.16	80.21 $\pm$ 1.28	79.96 $\pm$ 1.31	76.30 $\pm$ 5.51
	APPNP	83.68 $\pm$ 0.36	82.52 $\pm$ 0.46	83.45 $\pm$ 0.45	83.20 $\pm$ 0.53	81.33 $\pm$ 1.79
Citeseer	GCN	71.32 $\pm$ 0.70	71.93 $\pm$ 0.84	72.39 $\pm$ 0.66	71.79 $\pm$ 0.99	68.22 $\pm$ 4.13
	GAT	70.99 $\pm$ 0.60	71.78 $\pm$ 0.56	72.21 $\pm$ 0.52	71.81 $\pm$ 0.70	68.28 $\pm$ 2.63
	SGC	72.46 $\pm$ 0.15	74.13 $\pm$ 0.05	73.81 $\pm$ 0.12	73.32 $\pm$ 0.13	69.19 $\pm$ 2.08
	gfNN	67.33 $\pm$ 6.58	71.74 $\pm$ 1.22	71.98 $\pm$ 1.15	71.10 $\pm$ 1.22	64.63 $\pm$ 6.61
	APPNP	72.10 $\pm$ 0.38	72.94 $\pm$ 0.48	72.90 $\pm$ 0.48	72.63 $\pm$ 0.83	69.64 $\pm$ 2.29
Pubmed	GCN	79.23 $\pm$ 0.43	79.01 $\pm$ 0.55	79.03 $\pm$ 0.46	78.85 $\pm$ 0.67	76.99 $\pm$ 4.62
	GAT	79.05 $\pm$ 0.38	78.50 $\pm$ 0.61	78.85 $\pm$ 0.38	78.19 $\pm$ 0.56	78.00 $\pm$ 1.43
	SGC	78.72 $\pm$ 0.04	79.05 $\pm$ 0.08	79.30 $\pm$ 0.03	79.88 $\pm$ 0.19	77.83 $\pm$ 1.57
	gfNN	77.94 $\pm$ 2.32	77.92 $\pm$ 1.11	78.16 $\pm$ 0.98	77.76 $\pm$ 1.33	75.66 $\pm$ 3.05
	APPNP	80.09 $\pm$ 0.25	80.12 $\pm$ 0.44	80.15 $\pm$ 0.30	79.50 $\pm$ 0.47	78.37 $\pm$ 1.64

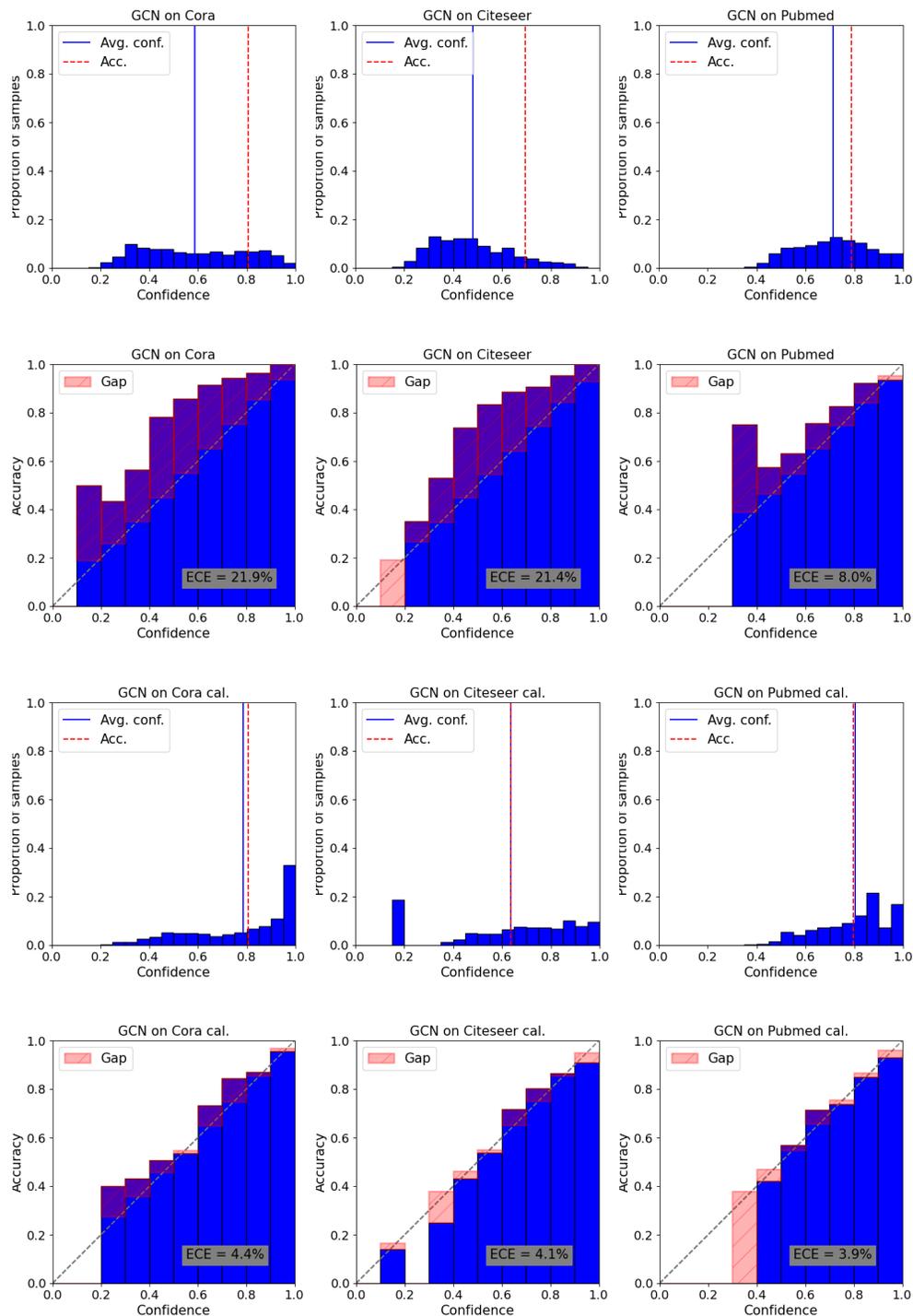


Fig. 8. Histograms and reliability diagrams for GCN.

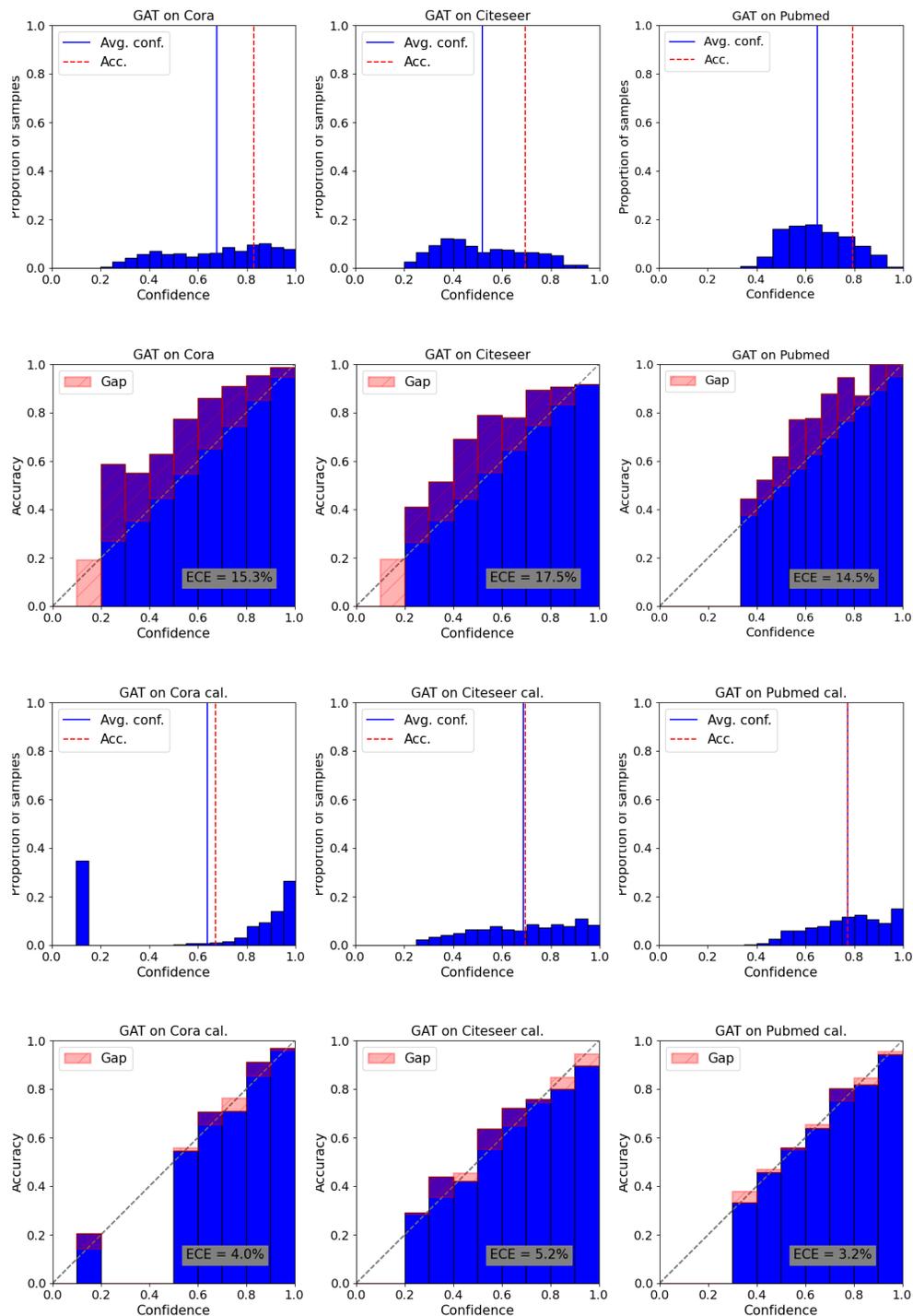


Fig. 9. Histograms and reliability diagrams for GAT.

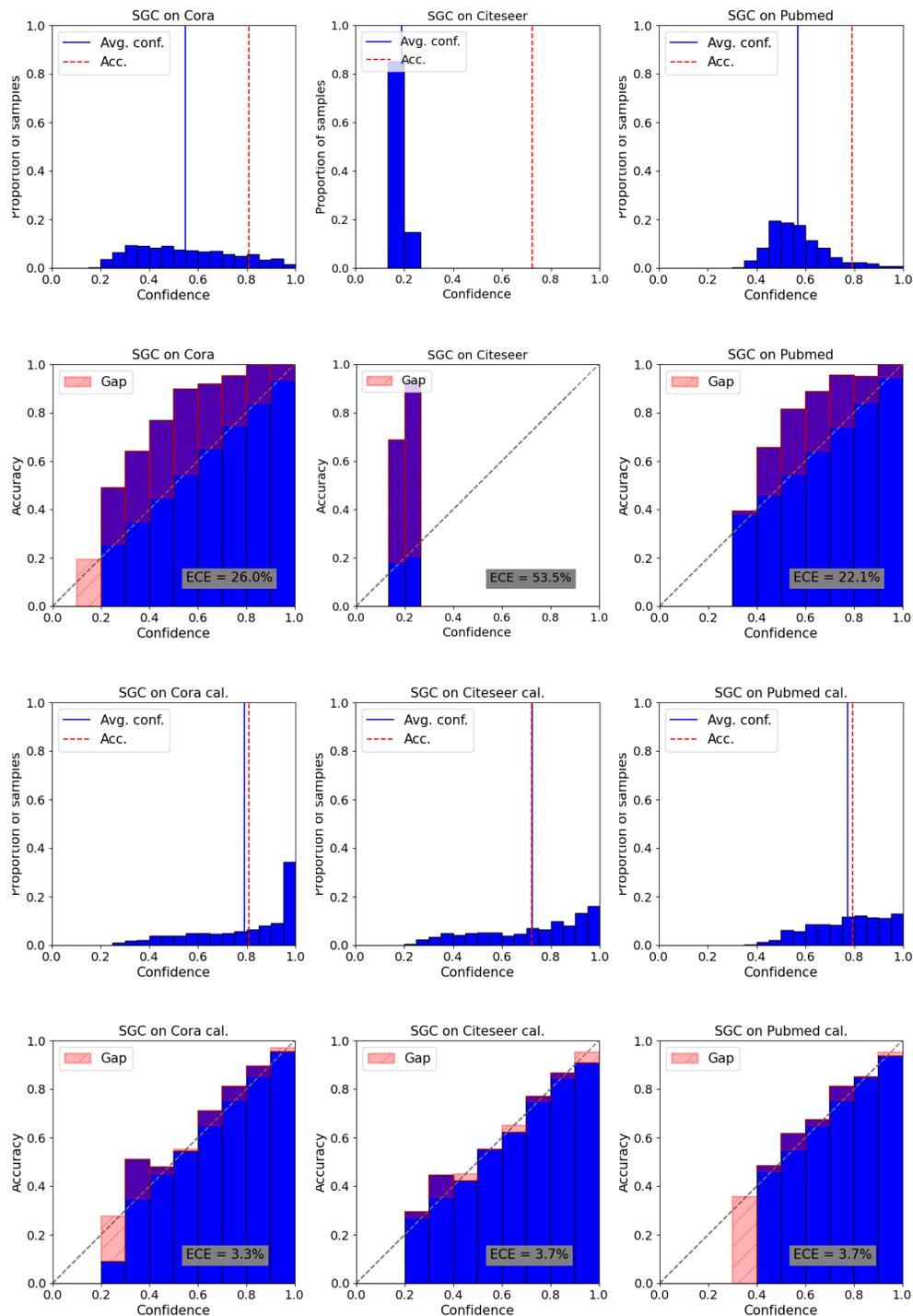


Fig. 10. Histograms and reliability diagrams for SGC.

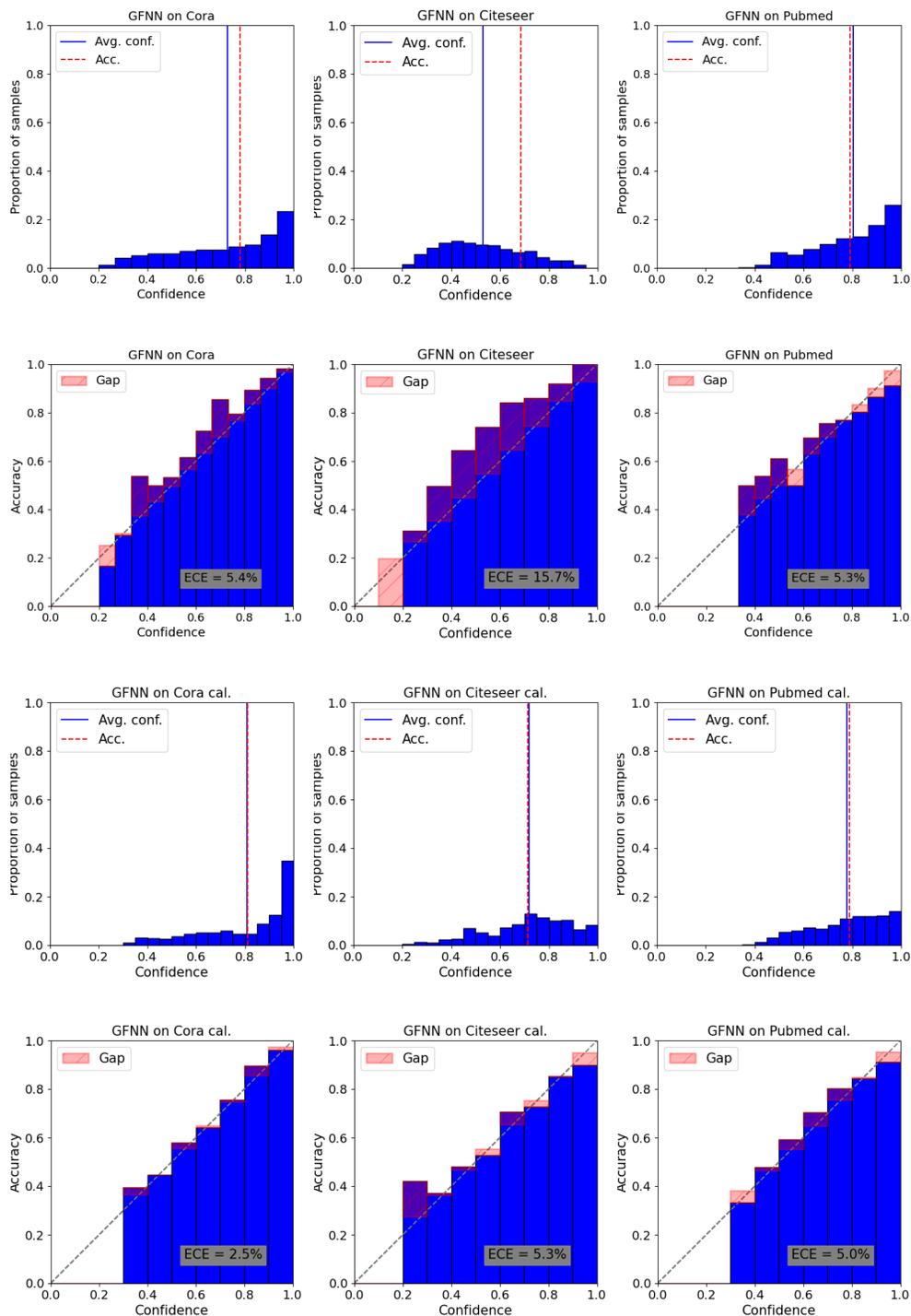


Fig. 11. Histograms and reliability diagrams for gfNN.

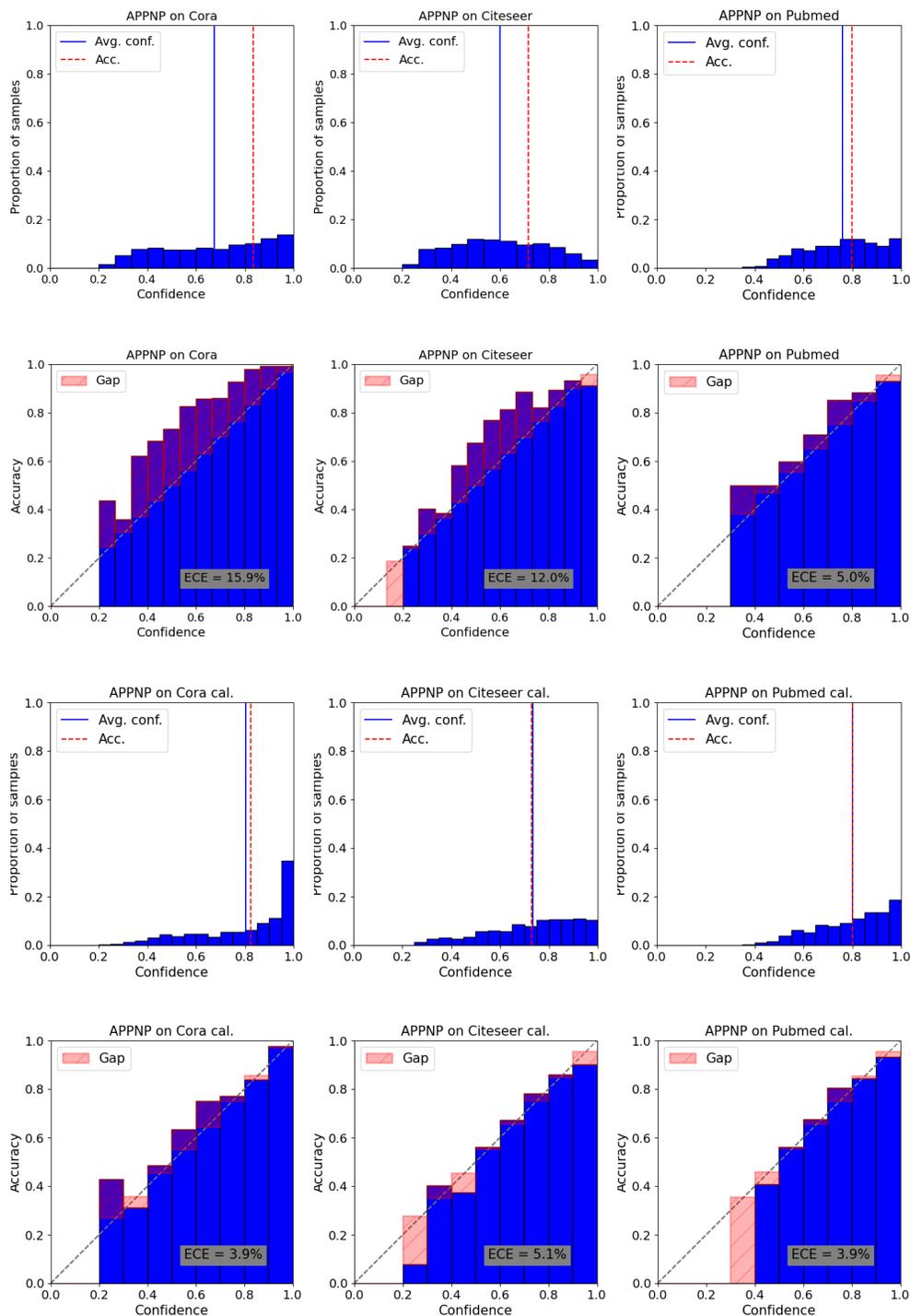


Fig. 12. Histograms and reliability diagrams for APPNP.

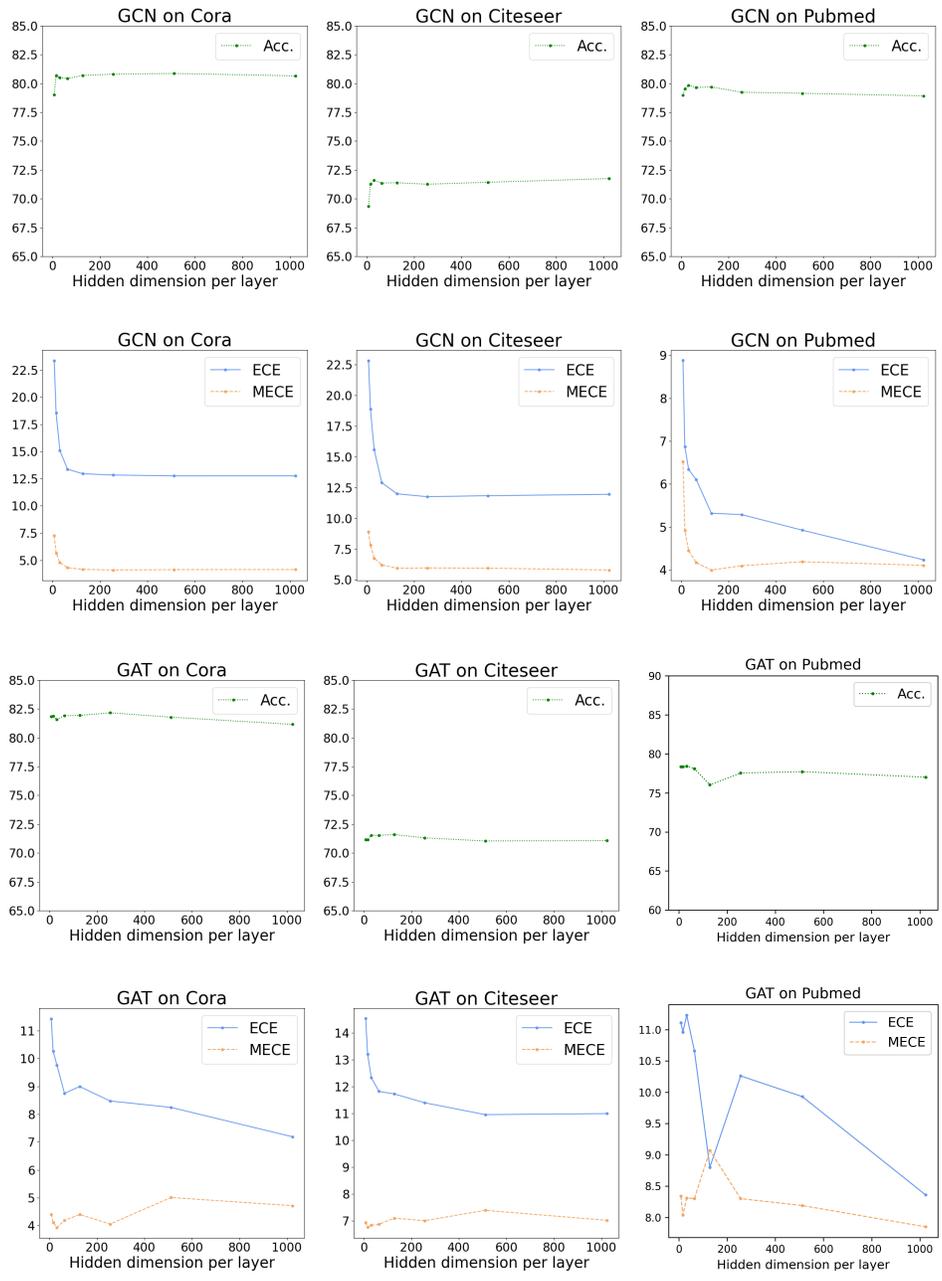


Fig. 13. Influence of width.

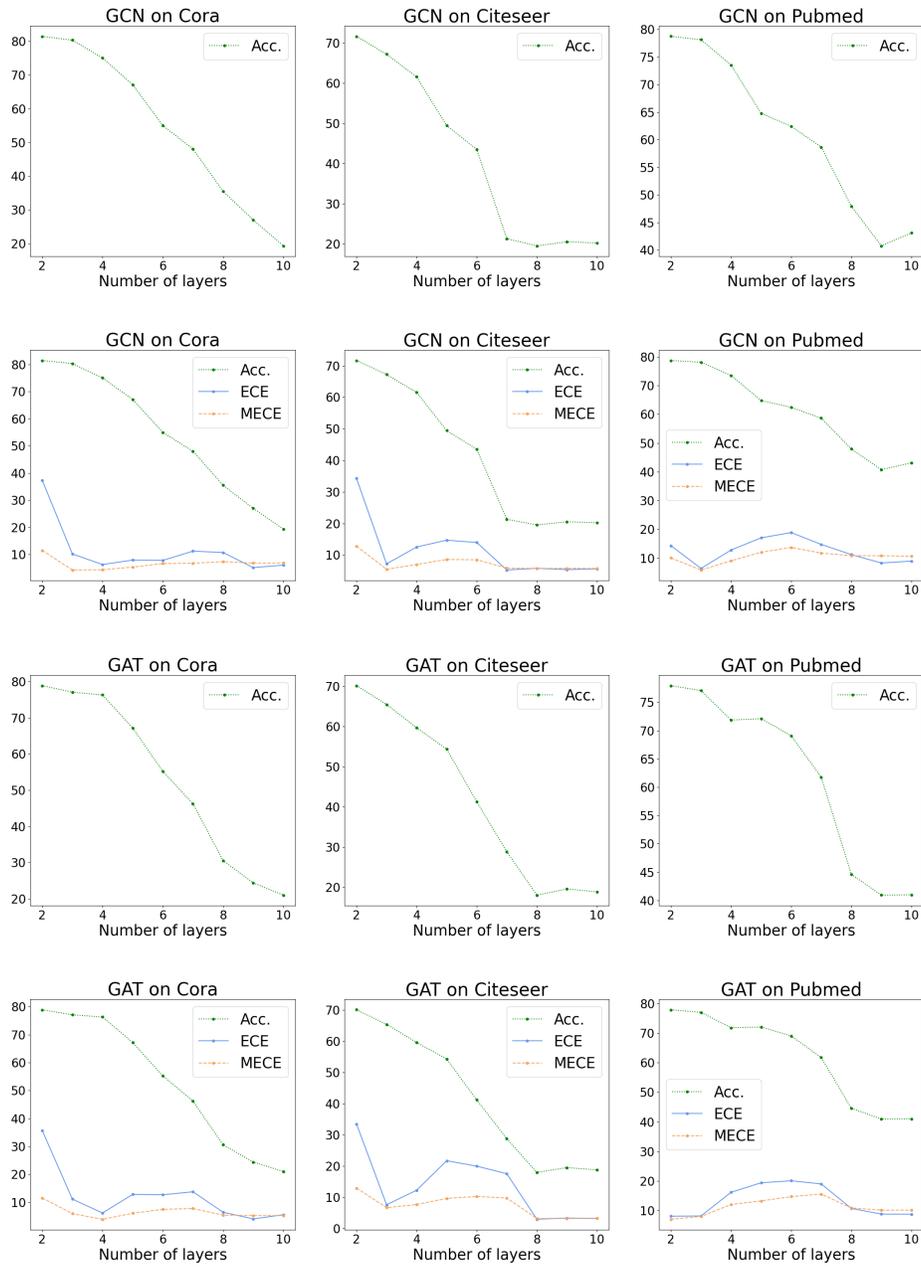


Fig. 14. Influence of depth.

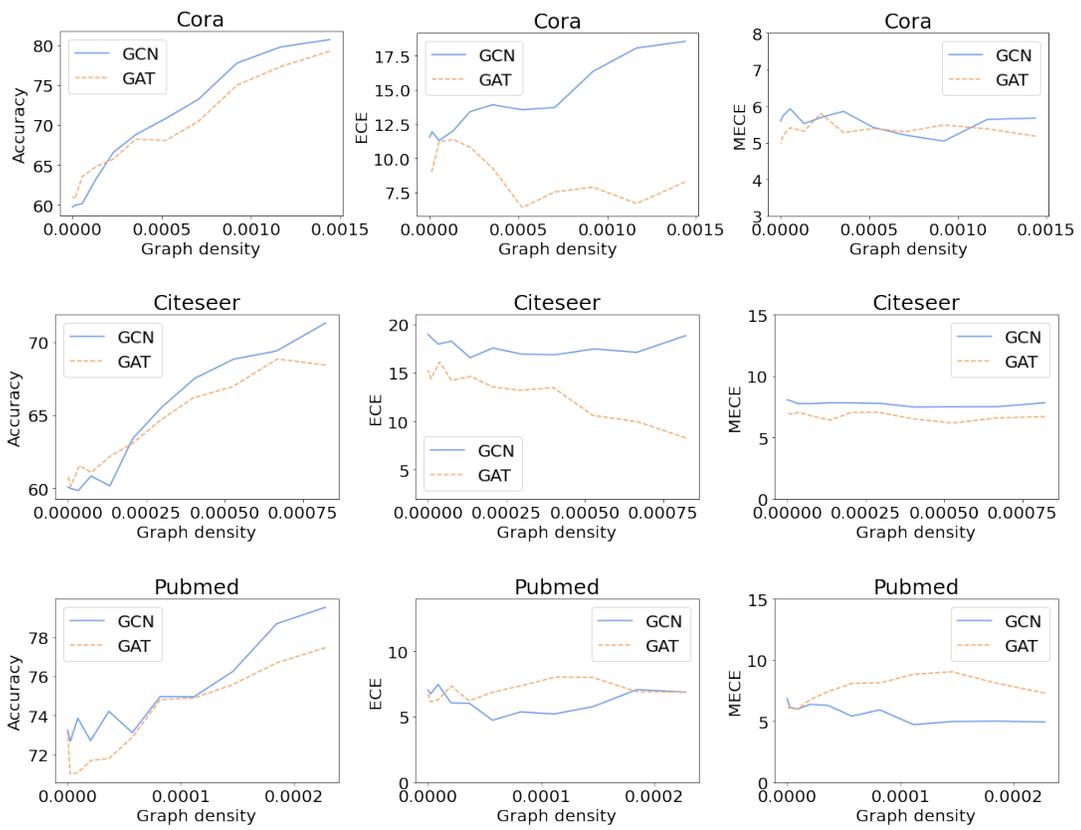


Fig. 15. Influence of graph density.