What Makes Graph Neural Networks Miscalibrated?

Hans Hao-Hsun Hsu[*1] **Yuesong Shen**[*1,2] **Christian Tomani** [1,2] **Daniel Cremers** [1,2]
[1] Technical University of Munich, Germany
[2] Munich Center for Machine Learning, Germany
{hans.hsu, yuesong.shen, christian.tomani, cremers}@tum.de

Abstract

Given the importance of getting calibrated predictions and reliable uncertainty estimations, various post-hoc calibration methods have been developed for neural networks on standard multi-class classification tasks. However, these methods are not well suited for calibrating graph neural networks (GNNs), which presents unique challenges such as accounting for the graph structure and the graph-induced correlations between the nodes. In this work, we conduct a systematic study on the calibration qualities of GNN node predictions. In particular, we identify five factors which influence the calibration of GNNs: general under-confident tendency, diversity of nodewise predictive distributions, distance to training nodes, relative confidence level, and neighborhood similarity. Furthermore, based on the insights from this study, we design a novel calibration method named Graph Attention Temperature Scaling (GATS), which is tailored for calibrating graph neural networks. GATS incorporates designs that address all the identified influential factors and produces nodewise temperature scaling using an attention-based architecture. GATS is accuracy-preserving, data-efficient, and expressive at the same time. Our experiments empirically verify the effectiveness of GATS, demonstrating that it can consistently achieve state-of-the-art calibration results on various graph datasets for different GNN backbones.[2]

1 Introduction

Graph-structured data, such as social networks, knowledge graphs and internet of things, have widespread presence and learning on graphs using neural networks has been an active area of research. For node classification on graphs, a wide range of graph neural network (GNN) models, including GCN [14], GAT [40] and GraphSAGE [9], have been proposed to achieve high classification accuracy.

This said, high accuracy is not the only desideratum for a classifier. Especially, reliable uncertainty estimation is crucial for applications like safety critical tasks and active learning. Neural networks are known to produce poorly calibrated predictions that are either overconfident or under-confident [7, 41]. To mitigate this issue a variety of post-hoc calibration methods [7, 19, 43, 38, 8] have been introduced over the last few years for calibrating neural networks on standard multi-class classification problems. However, calibration of GNNs, in the context of node classification on graphs, is currently still an underexplored topic. While it is possible to apply existing calibration methods designed for multi-class classification to GNNs in a nodewise manner, this does not address the specific challenges of node classification on graphs. Especially, node predictions in a graph are not i.i.d. but correlated, and we are tackling a structured prediction problem [25]. A uniform treatment when calibrating node predictions would fail to account for the structural information from graphs and the non i.i.d. behavior of node predictions.

[*]Equal contribution
[2]Source code available at `https://github.com/hans66hsu/GATS`

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Our contribution. In this work, we focus on calibrating GNNs for the node classification task [14, 40]. First, we aim at understanding the specific challenges posed by GNNs by conducting a systematic study on the calibration qualities of GNN node predictions. Our study reveals five factors that influence the calibration performance of GNNs: general under-confident tendency, diversity of nodewise predictive distributions, distance to training nodes, relative confidence level, and neighborhood similarity. Second, we develop Graph Attention Temperature Scaling (GATS) approach, which is designed in a way that accounts for the aforementioned influential factors. GATS generates nodewise temperatures that calibrate GNN predictions based on the graph topology. Third, we conduct a series of GNN calibration experiments and empirically verify the effectiveness of GATS in terms of calibration, data-efficiency, and expressivity.

2 Related work

For standard multi-class classification tasks, a variety of post-hoc calibration methods have been proposed in order to make neural networks uncertainty aware: temperature scaling (TS) [7], ensemble temperature scaling (ETS) [43], multi-class isotonic regression (IRM) [43], Dirichlet calibration [19], spline calibration [8], etc. Additionally, calibration has been formulated for regression tasks [17]. More generally, instead of transforming logits after training a classifier, a plethora of methods exists that modify either the model architecture or the training process itself. This includes methods that are based on Bayesian paradigm [12, 1, 6, 22, 42], evidential theory [33], adversarial calibration [37] and model ensembling [20]. One common caveat of these methods is the trade-off between accuracy and calibration, which oftentimes do not go hand in hand. Post-hoc methods like temperature scaling, on the other hand, are accuracy preserving. They ensure that the per node logit rankings are unaltered.

Calibration of GNNs is currently a substantially less explored topic. Nodewise post-hoc calibration on GNNs using methods developed for the multi-class setting has been empirically evaluated by Teixeira et al. [36]. They show that these methods, which perform uniform calibration of nodewise predictions, are unable to produce calibrated predictions for some harder tasks. Wang et al. [41] observe that GNNs tend to be under-confident in contrast to the majority of multi-class classifiers, which are generally overconfident [7]. Based on their findings, Wang et al. [41] propose the CaGCN approach, which attaches a GCN on top of the backbone GNN for calibration. Some approaches improve the uncertainty estimation of GNNs by adjusting model training. This includes Bayesian learning approaches [45, 10] and methods based on the evidential theory [46, 35].

3 Problem setup for GNN calibration

We consider the problem of calibrating GNNs for node classification tasks: given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the training data consist of nodewise input features $\{x_i\}_{i\in\mathcal{V}} \in \mathcal{X}$ and ground-truth labels $\{y_i\}_{i\in\mathcal{L}} \in \mathcal{Y} = \{1, \dots, K\}$ for a subset $\mathcal{L} \subset \mathcal{V}$ of nodes, and the goal is to predict the labels $\{y_i\}_{i\in\mathcal{U}} \in \mathcal{Y}$ for the rest of the nodes $\mathcal{U} = \mathcal{V} \setminus \mathcal{L}$. A graph neural network tackles the problem by producing nodewise probabilistic forecasts $\hat{p}_i$. These forecasts yield the corresponding label predictions $\hat{y}_i := \operatorname{argmax}_y \hat{p}_i(y)$ and confidences $\hat{c}_i := \max_y \hat{p}_i(y)$. The GNN is calibrated when its probabilistic forecasts are reliable, e.g., for predictions with confidence 0.8, they should be correct 80% of the time. Formally, a GNN is *perfectly calibrated* [41] if

$$\forall c \in [0, 1], \quad \mathbb{P}(y_i = \hat{y}_i | \hat{c}_i = c) = c. \tag{1}$$

In practice, we quantify the calibration quality with the expected calibration error (ECE) [27, 7]. We follow the commonly used definition from Guo et al. [7] which uses a equal width binning scheme to estimate calibration error for any node subset $\mathcal{N} \subset \mathcal{V}$: the predictions are regrouped according to M equally spaced confidence intervals, i.e. $(B_1, \dots, B_M)$ with $B_m = \{j \in \mathcal{N} \mid \frac{m-1}{M} < \hat{c}_j \leq \frac{m}{M}\}$, and the expected calibration error of the GNN forecasts is defined as

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{|\mathcal{N}|} \Big|\text{acc}(B_m) - \text{conf}(B_m)\Big|, \text{ with} \tag{2}$$

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \mathbf{1}(y_i = \hat{y}_i) \text{ and } \text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \hat{c}_i. \tag{3}$$

4 Factors that influence GNN calibration

To design calibration methods adapted to GNNs, we need to figure out the particular factors that influence the calibration quality of GNN predictions. For this we train a series of GCN [14] and GAT [40] models on seven graph datasets: Cora [32], Citeseer [32], Pubmed [23], Amazon Computers [34], Amazon Photo [34], Coauthor CS [34], and Coauthor Physics [34]. We summarize the dataset statistics in Appendix A.1. Details about model training are provided in Appendix A.2 for reproducibility. To compare with the standard multi-class classification case, we additionally train ResNet-20 [11] models on the CIFAR-10 image classification task [16] as a reference.

Our experiments uncover five decisive factors that affect the calibration quality of GNNs. In the following we discuss them in detail.

4.1 General under-confident tendency

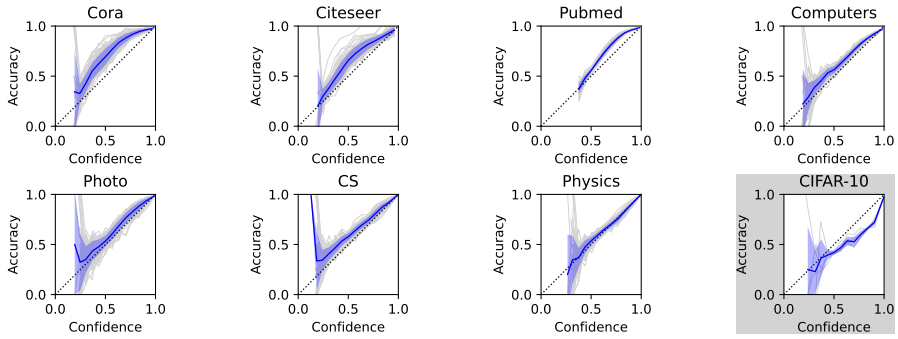

Figure 1: Reliability diagrams of GCN models trained on various graph datasets. We see a general tendency of under-confident predictions (plots above the diagonal) except the Physics dataset. This is in contrast to the overconfident behavior of multi-class image classification using CNNs (in gray).

Starting with a global perspective, we notice that GNNs tend to produce under-confident predictions. In Figure 1 we plot the reliability diagrams [24] for results on different graph datasets using GCN. Similar to Wang et al. [41], we see a general trend of under-confident predictions for GNNs. This is in contrast to the standard multi-class image classification case which has overconfident behavior. Also, it is interesting to see that this under-confident trend can be more or less pronounced depending on the dataset. For Coauthor Physics, the predictions are well calibrated and have no significant bias.

Results using GAT models lead to similar conclusions and are provided in Appendix B.1.

4.2 Diversity of nodewise predictive distributions

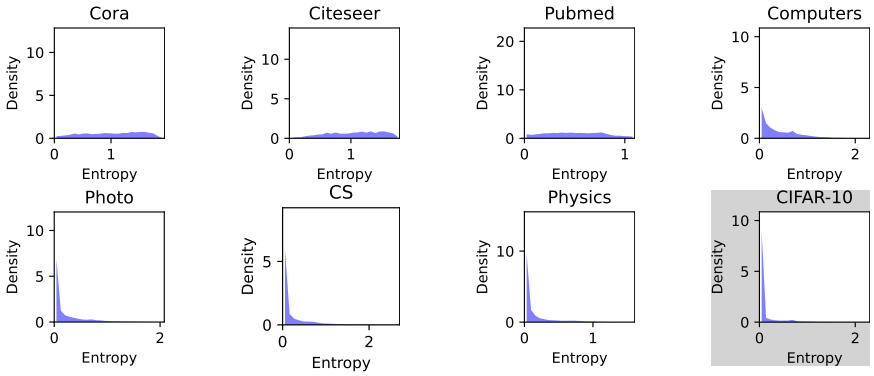

Figure 2: Entropy distributions of GCN predictions on graph datasets. Compared to the standard classification case, GNN predictions tend to be more dispersed, reflecting their disparate behaviors.

Contrary to the standard multi-class case, GNN outputs can have varying roles depending on their positions in the graph, which means that their output distributions could exhibit dissimilar behaviors. This is empirically evident in Figure 2, where we visualize the entropy distributions of GCN output predictions v.s. the standard multi-class results (GAT results are available in Appendix B.2). We see that the entropies of GNN outputs have more spread-out distributions, which indicates that they have distinct roles and behaviors in graphs.

In terms of GNN calibration, this observation implies that uniform node-agnostic adjustments like temperature scaling [7] might be insufficient for GNNs, whereas nodewise adaptive approaches could be beneficial.

4.3 Distance to training nodes

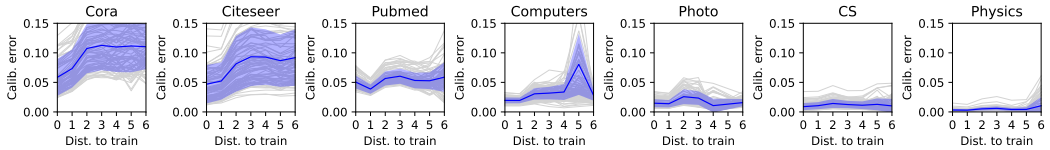

Figure 3: Nodewise calibration error of GCN results depending on the minimum distance to training nodes. We observe that training nodes and their neighbors tend to be better calibrated.

A graph provides additional structural information for its nodes. One insightful feature is the minimum distance to training nodes. We discover that nodes with shorter distances, especially the training nodes themselves and their direct neighbors, tend to be better calibrated.

To evaluate the calibration quality nodewise, we propose the *nodewise calibration error*, which is based on the binning scheme used to compute the global expected calibration error (ECE) [27, 7]: for each node, we find its corresponding bin depending on its predicted confidence, and the calibration error of this bin is assigned to be its nodewise calibration error.

Using this nodewise metric, in Figure 3 we visualize the influence of minimum distance to training nodes on the nodewise calibration quality (c.f. Appendix B.3 for GAT results). We see that nodes close to training ones typically have lower nodewise calibration error. This suggests that minimum distance to training nodes can be useful for GNN calibration.

4.4 Relative confidence level

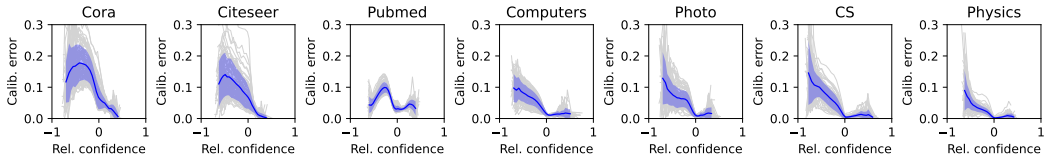

Figure 4: Nodewise calibration error of GCN results depending on the relative confidence level. We observe that nodes which are less confident than their neighbors tend to have worse calibration.

Another important structural information is the neighborhood relation. We find out that the relative confidence level $\delta\hat{c}_i$ of a node i, i.e., the difference between the nodewise confidence $\hat{c}_i$ and the average confidence of its neighbors

$$\delta\hat{c}_i = \hat{c}_i - \frac{1}{|n(i)|} \sum_{j \in n(i)} \hat{c}_j \tag{4}$$

has an interesting correlation to the nodewise calibration quality. In Figure 4 we show the relation between the relative confidence level of a node and its nodewise calibration error (c.f. Appendix B.4 for GAT results). Especially, We observe that nodes which are less confident than their neighbors tend to have worse calibration, and it is in general desirable to have comparable confidence level w.r.t. the neighbors. For GNN calibration, the relative confidence level $\delta\hat{c}_i$ can be a useful node feature to consider.

4.5 Neighborhood similarity

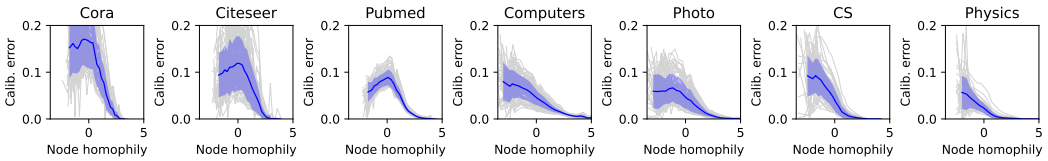

Figure 5: Nodewise calibration error of GCN results depending on the node homophily. Nodes with strongly agreeing neighbors tend to have significantly lower calibration errors.

Furthermore, we find that different neighbors tend to introduce distinct influences. For assortative graphs which are the focus of this work, we find out that calibration of nodes are affected by *node homophily*, i.e., whether a node tends to have the same label prediction as its neighbors. For a node with n_a agreeing neighbors and n_d disagreeing ones, we measure the node homophily as

$$\text{Node homophily} = \log\left(\frac{n_a + 1}{n_d + 1}\right), \tag{5}$$

where positive values indicate greater ratio of agree neighbors and vice versa.

Figure 5 summarizes the variation of nodewise calibration error w.r.t. the node homophily for different graph datasets (c.f. Appendix B.5 for GAT results). We find out that nodewise calibration errors tend to decrease significantly for nodes with strongly agreeing neighbors. This suggests that neighborhood predictive similarity should be considered when doing GNN calibration.

5 Graph attention temperature scaling (GATS)

Based on the findings in Section 4, we design a new post-hoc calibration method, named *Graph Attention Temperature Scaling (GATS)*, which is tailored for GNNs.

5.1 Formulation and design of GATS

To obtain a calibration method that is adapted to the graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and reflects the observed influential factors in Section 4, the graph attention temperature scaling approach extends the temperature scaling [7] method to produce a distinct temperature T_i for each node $i \in \mathcal{V}$. T_i is then used to scale the uncalibrated nodewise output logits z_i and produce calibrated node predictions $\hat{p}_i$

$$\forall i \in \mathcal{V}, \quad \hat{p}_i = \text{softmax}\left(\frac{z_i}{T_i}\right). \tag{6}$$

Formulation of T_i. The nodewise temperature T_i should address the five factors discussed in Section 4. We achieve this via the following considerations:

- We introduce a global bias parameter T_0 to account for the general under-confident tendency;
- To tackle the diverse behavior of node predictions, we learn a nodewise temperature contribution τ_i based on the predicted nodewise logits z_i;
- To incorporate the relative confidence w.r.t. neighbors, we introduce $\delta\hat{c}_i$ from Eq. 4 as an additional contribution term scaled by a learnable coefficient ω;
- To model the influence of neighborhood similarity, we use an attention mechanism [39] to aggregate neighboring contributions τ_j with attention coefficients $\alpha_{i,j}$ depending on the output similarities between the neighbors i and j;
- Distance to training nodes is used to introduce a nodewise scaling factor γ_i to adjust the node contribution and the aggregation process. It is learnable for training nodes and their direct neighbors and fixed to 1 for the rest:

$$\gamma_i = \begin{cases} \gamma_t, & \text{if } i \text{ is a training node} \\ \gamma_n, & \text{if } i \text{ is a neighbor of training node} \\ 1, & \text{otherwise} \end{cases}, \quad \gamma_t, \gamma_n \text{ learnable parameters.} \tag{7}$$

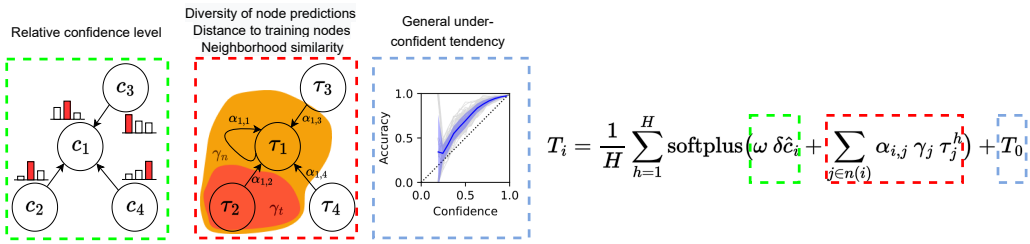

Figure 6: Illustration of graph attention temperature scaling (GATS) for a graph with four nodes where node 2 is a training node and node 1 is the target node which aggregates information from the other three nodes. GATS incorporates the five factors discussed in Section 4 and produces nodewise temperatures T_i using the graph structure and an attention mechanism.

Putting together the above components, the nodewise temperature T_i has the following expression:

$$\forall i \in \mathcal{V}, \quad T_i = \frac{1}{H}\sum_{h=1}^{H} \text{softplus}\left(\omega\ \delta\hat{c}_i + \sum_{j\in\hat{n}(i)} \alpha_{i,j}\ \gamma_j\ \tau_j^h\right) + T_0. \tag{8}$$

Here we have a multi-head formulation where h indicates the h-th attention head, and $\hat{n}(i)$ denotes the neighbors of node i including self-loop. We uses 8 heads ($H = 8$), which works well in practice. (c.f. Section 6.3.)

Defining τ_i^h. Nodewise contribution τ_i^h are computed as the outputs of parameterized linear layers $\phi^h(\cdot;\theta)$ which take transformed nodewise output logits $\tilde{z}_i$ as input

$$\forall i \in \mathcal{V}, \forall h \in [1, H], \quad \tau_i^h = \phi^h(\tilde{z}_i;\theta^h) = (\theta^h)^\top\ \tilde{z}_i. \tag{9}$$

The transformed nodewise output logits $\tilde{z}_i$ are produced as follows: we first normalize the original logits z_i to range $[0, 1]$, then sort the classwise logits for each node. This makes the linear layers ϕ^h focus on the general logit distributions rather than class predictions. A similar idea has been explored by Rahimi et al. [30], where they show that intra order-preserving functions improve the model calibration. Here we find out that this sorting-based transformation helps GATS to learn useful representations of nodewise contributions τ_i^h. (c.f. Section 6.3.)

Defining $\alpha_{i,j}$. The attention coefficients $\alpha_{i,j}$ are defined based on the neighbor similarity, which is determined by the inner product between rescaled nodewise logits z_i/γ_i and z_j/γ_j. Inspired by Veličković et al. [40], we compute the attention coefficients $\alpha_{i,j}$ as follows:

$$(\alpha_{i,j})_{j\in\hat{n}(i)} = \underset{j\in\hat{n}(i)}{\text{softmax}}\left(\text{leakyReLU}\left(\frac{1}{\gamma_i\ \gamma_j} z_i^\top\ z_j\right)\right). \tag{10}$$

5.2 Calibration properties of GATS

Zhang et al. [43] propose three desiderata for calibration methods: accuracy-preserving, data-efficient, and expressive. GATS fulfills all of them:

- GATS is accuracy-preserving: since all node predictions are scaled by inverse temperatures $1/T_i$ which are positive scalars, the order of output logits is preserved;
- GATS is data-efficient. It is a parametric calibration model with $C \cdot H + 4$ learnable parameters $(T_0, \omega, \gamma_t, \gamma_n, (\theta^h)_{1\leq h\leq H})$. According to Zhang et al. [43], parametric methods are already data-efficient;
- GATS is expressive, as it produces nodewise temperatures T_i adapted to the graph structure.

Experiments in Section 6.2 empirically confirm the data-efficiency and expressivity of GATS.

5.3 Comparison with CaGCN

It is interesting to compare our proposed GATS approach to the CaGCN method proposed by Wang et al. [41], since both approaches aim at calibrating GNNs, and both make use of the graph structure to produce nodewise temperatures. While CaGCN uses a GCN to generate nodewise temperatures straightforwardly, GATS uses an attention mechanism which differentiates the influence from various neighbors. GATS also integrates a series of careful designs following the insights from the study in Section 4. Experiments in Section 6.1 shows that GATS tends to produce better calibration results compared to CaGCN.

6 Experiments

To evaluate the performance of GATS on GNN calibration and understand the effects of its designs, we conduct a series of experiments for baseline comparison and ablation study. We use two representative GNNs: GCN and GAT, which are trained on the seven aforementioned graph datasets plus a larger graph, CoraFull [2], for post-hoc calibration. We display the ECE results with $M = 15$ bins. We use the expected calibration error (ECE) [27, 7] with $M = 15$ bins as an evaluation metric, and follow an experimental protocol similar to Kull et al. [19, 18]: For all the experiments, we randomly split the labeled/unlabeled (15%/85%) data five times, and use three-fold internal cross-validation of the labeled data to train the GNNs and the calibrators. We also utilize five random initializations, resulting in 75 total runs for each experiment. We provide detailed experimental settings in Appendix A.

6.1 Performance comparison

We benchmark GATS against existing baselines on a variety of GNN calibration tasks. We compare GATS with the following baselines:

- **Temperature scaling (TS)** [7] simply uses a global temperature to scale the logits.
- **Vector scaling (VS)** [7] scales the logits separately over the class dimension and additionally introduces a classwise bias for the recalibrated output logits.
- **Ensemble temperature scaling (ETS)** [43] learns a mixture of uncalibrated, TS-calibrated, and uniform probabilistic outputs.
- **GCN as a calibration function (CaGCN)** [41] is specifically designed for calibrating GNNs. It uses a GCN on top to generate nodewise temperatures.

Additionally, we also report the ECEs of uncalibrated predictions as a reference. Among the above baselines, TS, VS, and ETS are calibration methods designed for standard classification cases and operate on nodes uniformly. CaGCN on the other hand performs separate nodewise adjustments and uses the graph structure, similar to our proposed GATS approach.

For the post-hoc calibration experiments, we fix the weight of the trained GNN backbones and adjust the parameters of the calibration methods on the validation set. Negative log-likelihood is chosen as the objective for the calibration process. We provide details of method configurations and calibration settings in Appendix A.3. Table 1 summarizes the calibration results.

Overall, we observe that GATS consistently produces well-calibrated predictions for all graph datasets and GNN backbones. Except for the GAT model trained on Pubmed (3rd best) and the GCN model trained on Amazon Computers (2nd best), GATS achieves the highest calibration quality in all cases.

Also, it is interesting to see that for all cases the best result is achieved by methods which use the graph structure and produce adapted adjustments for different nodes. This demonstrates the necessity of designing calibration methods that address the unique challenges posed by GNN calibration.

Although CaGCN can get the best results for Pubmed using GAT and Amazon Computers using GCN, we see that its performance is rather unstable for different scenarios, and sometimes it can even produce worse calibration results than the uncalibrated baseline. Using their proposed margin-based loss did not help in our settings. We suspect that CaGCN might have an overly complex architecture for the task, and it cannot differentiate neighborhood influences with the common normalized adjacency matrix. Our proposed GATS model does not have this issue. It has consistent and good calibration performance in all cases.

Table 1: GNN calibration results in terms of ECE (in percentage, lower is better) of GATS and other baseline methods on various graph datasets. Overall, GATS achieves state-of-the-art performance, getting the best results in most scenarios. Also, all best results are achieved by methods that consider the graph structure. This shows the need for dedicated methods to tackle GNN calibration.

Dataset	Model	Uncal	TS	VS	ETS	CaGCN	GATS
Cora	GCN	13.04±5.22	3.92±1.29	4.36±1.34	3.79±1.35	5.29±1.47	**3.64±1.34**
	GAT	23.31±1.81	3.69±0.90	3.30±1.12	3.54±1.01	4.09±1.06	**3.18±0.90**
Citeseer	GCN	10.66±5.92	5.15±1.50	4.92±1.44	4.65±1.69	6.86±1.41	**4.43±1.30**
	GAT	22.88±3.53	4.74±1.47	4.25±1.48	4.11±1.64	5.75±1.31	**3.86±1.56**
Pubmed	GCN	7.18±1.51	1.26±0.28	1.46±0.29	1.24±0.30	1.09±0.52	**0.98±0.30**
	GAT	12.32±0.80	1.19±0.36	1.00±0.32	1.20±0.32	**0.98±0.31**	1.03±0.32
Computers	GCN	3.00±0.80	2.65±0.57	2.70±0.63	2.58±0.70	**1.72±0.53**	2.23±0.49
	GAT	1.88±0.82	1.63±0.46	1.67±0.52	1.54±0.67	2.03±0.80	**1.39±0.39**
Photo	GCN	2.24±1.03	1.68±0.63	1.75±0.63	1.68±0.89	1.99±0.56	**1.51±0.52**
	GAT	2.02±1.11	1.61±0.63	1.63±0.69	1.67±0.73	2.10±0.78	**1.48±0.61**
CS	GCN	1.65±0.92	0.98±0.27	0.96±0.30	0.94±0.24	2.27±1.07	**0.88±0.30**
	GAT	1.40±1.25	0.93±0.34	0.87±0.35	0.88±0.33	2.52±1.04	**0.81±0.30**
Physics	GCN	0.52±0.29	0.51±0.19	0.48±0.16	0.52±0.19	0.94±0.51	**0.46±0.16**
	GAT	0.45±0.21	0.50±0.21	0.52±0.20	0.50±0.21	1.17±0.42	**0.42±0.14**
CoraFull	GCN	6.50±1.26	5.54±0.43	5.76±0.42	5.38±0.49	5.86±2.52	**3.76±0.74**
	GAT	4.73±1.39	4.00±0.50	4.17±0.43	3.89±0.56	6.55±3.69	**3.54±0.63**

We also observe that the results tend to have high variations, since GNN backbones tend to predict highly varying results when trained with different initial weights and random splits [34]. We ensure the reliability of the results by averaging over a total of 75 runs with various initial weights and random splits for each case.

6.2 Data-efficiency and expressivity of GATS

Furthermore, we analyze the data-efficiency and the expressivity of GATS for GNN calibration. For this we reuse the GNN models trained on the CoraFull dataset, and consider the influence of calibration sample size on the GNN calibration performance. For comparison we also report the corresponding results using ensemble temperature scaling and CaGCN. Figure 7 visualizes the results with GCN backbone. The results for GAT backbone are summarized in Appendix D.

Overall, we see that GATS is both data-efficient and expressive. It requires few calibration samples to get decent calibration performance. This is in contrast to CaGCN which needs more than 5% of nodes for calibration to get acceptable results. Compared to ETS, GATS is more expressive and has a considerably lower calibration error for CoraFull, which is a large graph dataset.

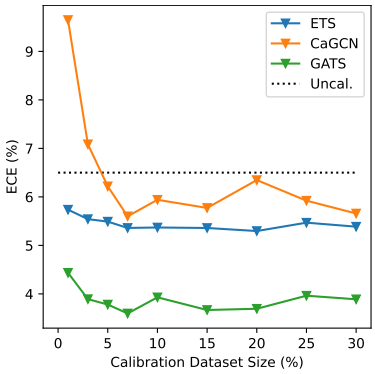

Figure 7: ECEs (in percentage) on CoraFull for ETS, CaGCN, and GATS using various amounts of calibration data. We see that GATS is data-efficient and expressive for GNN calibration.

6.3 Ablation study

To empirically analyze the effect of various GATS design choices, we conduct a series of ablation study experiments in this section. Overall, we notice that all designs are advantageous and removing any of them leads to a general decrease in performance.

Table 2: Ablation study results in terms of ECE (in percentage) for various GATS designs. Overall, all designs are beneficial and removing any of them leads to worse results in general.

Dataset	Model	w/o T_0	w/o γ_i	w/o $\delta\hat{c}_i$	w/o attention	w/o sorting	GATS
Cora	GCN	3.72±1.20	3.80±1.51	3.71±1.18	3.63±1.48	4.35±1.77	3.64±1.34
	GAT	3.25±1.00	3.46±1.00	3.24±0.89	3.69±0.96	4.18±1.70	3.18±0.90
Citeseer	GCN	5.50±1.76	4.73±1.45	4.49±1.30	4.95±1.56	5.87±1.99	4.43±1.30
	GAT	3.56±1.73	4.39±1.46	3.87±1.55	4.81±1.54	4.99±2.34	3.86±1.56
Photo	GCN	2.20±0.88	1.60±0.64	1.54±0.52	1.59±0.67	1.68±0.61	1.51±0.52
	GAT	2.37±1.01	1.53±0.63	1.47±0.62	1.62±0.66	1.77±0.70	1.48±0.61

Effect of global bias T_0. We consider the effect of global bias T_0 by comparing to a GATS variant without it (i.e., setting $T_0 = 0$ in Eq. (8)). Its results are collected in column "w/o T_0" of Table 2. Overall we see that a learnable bias T_0 is beneficial in most cases.

Effect of nodewise scaling factor γ_i. The nodewise scaling factors γ_i provide custom adjustment for training nodes and their neighbors. Removing its influence can be done by fixing all γ_i to one in Eqs. (8) and (10). The results of the variant without γ_i are recorded in column "w/o γ_i" of Table 2. They are worse in general, suggesting that nodewise scaling factors γ_i are indeed helpful.

Effect of relative confidence level $\delta\hat{c}_i$. To evaluate the impact of introducing nodewise relative confidence level $\delta\hat{c}_i$, we create a GATS variant where ω in Eq. (8) is fixed to zero, which effectively removes the influence of the relative confidence level $\delta\hat{c}_i$. In column "w/o $\delta\hat{c}_i$" of Table 2 we have the results corresponding to this variant, which is in general slightly worse than the standard GATS which includes $\delta\hat{c}_i$.

Effect of attention-based aggregation. To understand the role played by the attention mechanism, we create a GATS variant which completely removes the attention related term ($\sum_{j\in\hat{n}(i)} \alpha_{i,j}\ \gamma_j\ \tau_j^h$) from Eq. (8). Its performance is shown in column "w/o attention" of Table 2. Again, we observe that removing the attention-based component tends to worsen the calibration performance.

Effect of logit sorting. GATS uses normalized and sorted logits $\tilde{z}_i$ as input to generate nodewise temperature contributions τ_i^h. And we find that this sorting transform is essential for learning good representations of τ_i^h. In column "w/o sorting" of Table 2 we have the results for GATS variants which uses the logits without sorting to compute τ_i^h. And we observe that this deteriorates the calibration performance.

Table 3: Calibration results (ECE in percentage) of GATS models with different numbers of attention heads. 8 attention heads are sufficient for optimal GNN calibration results.

Dataset	Model	Number of Heads				
		1	2	4	8	16
Cora	GCN	3.95±1.45	3.75±1.42	3.79±1.45	3.66±1.33	3.50±1.25
	GAT	3.48±1.22	3.49±1.18	3.43±1.15	3.20±0.90	3.31±0.74
Citeseer	GCN	4.74±1.43	4.77±1.66	4.70±1.52	4.43±1.30	4.42±1.04
	GAT	4.53±1.66	4.57±1.55	4.20±1.50	3.86±1.56	4.20±1.54
Photo	GCN	1.51±0.54	1.54±0.59	1.52±0.58	1.51±0.52	1.50±0.51
	GAT	1.52±0.61	1.52±0.65	1.53±0.60	1.48±0.61	1.50±0.60

Effect of attention head count H. Finally, we analyze the influence of multi-head count H on the GNN calibration results. For this we run a series of experiments using GATS models with 1, 2, 4, 8, and 16 attention heads. The results are collected in Table 3. We see that for GCN backbones, GATS models with more attention heads tend to get better results. However, for GAT backbones, using 16 heads results in worse performance compared to 8 heads. Accounting also for the fact that doubling

the attention head count effectively doubles the computational requirements, 8 attention heads is a decent general setting for GATS.

7 Conclusion

In this work, we tackle the GNN calibration problem. We conduct a systematic study to analyze the calibration properties of GNNs predictions. Our study reveals five influential factors and manifests the unique challenges raised by GNN calibration. Based on the insights from our studies, we propose a novel calibrator, GATS, which accounts for the identified factors and is tailored for calibrating GNNs. GATS is accuracy-preserving, data-efficient, and expressive at the same time. Our experiments demonstrate that GATS achieves state-of-the-art performance for GNN calibration on various graph datasets and for different GNN backbones.

Our work focuses on the node classification tasks for assortative graphs, where neighbors tend to agree with each other. It is thus important to realize that the validity of the conclusions from Section 4 is limited to the assortative case, and might no longer hold for disassortative graphs [47, 29]. It can be an interesting future work to conduct similar studies for GNN calibration in the heterophilous case, especially when more established GNN architectures are available. More generally, devising calibration methodologies for other graph learning tasks such as link prediction [44] and graph classification [4] could also be an interesting direction for future research.

Acknowledgments and Disclosure of Funding

This work was supported by the Munich Center for Machine Learning (MCML) and by the ERC Advanced Grant SIMULACRON. The authors would like to thank Nikita Araslanov for proofreading and helpful discussions, as well as the anonymous reviewers for their constructive feedback.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *ICML*, 2015.

[2] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[3] P. de Haan. On the use of density kernels for concentration estimations within particle and puff dispersion models. *Atmospheric Environment*, 33(13):2007–2021, 1999. ISSN 1352-2310.

[4] F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. In *ICLR*, 2020.

[5] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[6] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[7] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[8] K. Gupta, A. Rahimi, T. Ajanthan, T. Mensink, C. Sminchisescu, and R. Hartley. Calibration of neural networks using splines. In *ICLR*, 2021.

[9] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[10] A. Hasanzadeh, E. Hajiramezanali, S. Boluki, M. Zhou, N. Duffield, K. Narayanan, and X. Qian. Bayesian graph neural networks with adaptive connection sampling. In *ICML*, 2020.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[12] J. M. Hernandez-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[15] D. Kraft. A software package for sequential quadratic programming. Technical report, DLR German Aerospace Center, 1988.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[17] V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. In *ICML*, 2018.

[18] M. Kull, T. M. S. Filho, and P. Flach. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. *Electronic Journal of Statistics*, 11(2):5052 – 5080, 2017.

[19] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *NeurIPS*, 2019.

[20] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

[21] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *MATHEMATICAL PROGRAMMING*, 45:503–528, 1989.

[22] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019.

[23] G. Namata, B. London, L. Getoor, and B. Huang. Query-driven active surveying for collective classification. In *ICML Workshop*, 2012.

[24] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *ICML*, 2005.

[25] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

[26] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*, 2018.

[27] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[29] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*, 2020.

[30] A. Rahimi, A. Shaban, C. Cheng, R. Hartley, and B. Boots. Intra order-preserving functions for calibration of multi-class neural networks. In *NeurIPS*, 2020.

[31] D. W. Scott. Multivariate density estimation. *Wiley*, 1992.

[32] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

[33] M. Sensoy, L. M. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. In *NeurIPS*, 2018.

[34] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[35] M. Stadler, B. Charpentier, S. Geisler, D. Zügner, and S. Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *NeurIPS*, 2021.

[36] L. Teixeira, B. Jalaian, and B. Ribeiro. Are graph neural networks miscalibrated? In *ICML Workshop*, 2019.

[37] C. Tomani and F. Buettner. Towards trustworthy predictions from deep neural networks with fast adversarial calibration. In *AAAI*, 2021.

[38] C. Tomani, S. Gruber, M. E. Erdem, D. Cremers, and F. Buettner. Post-hoc uncertainty calibration for domain drift scenarios. In *CVPR*, 2021.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[41] X. Wang, H. Liu, C. Shi, and C. Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. In *NeurIPS*, 2021.

[42] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[43] J. Zhang, B. Kailkhura, and T. Y. Han. Mix-n-match : Ensemble and compositional methods for uncertainty calibration in deep learning. In *ICML*, 2020.

[44] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *NeurIPS*, 2018.

[45] Y. Zhang, S. Pal, M. Coates, and D. Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI*, 2019.

[46] X. Zhao, F. Chen, S. Hu, and J. Cho. Uncertainty aware semi-supervised learning on graph data. In *NeurIPS*, 2020.

[47] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *NeurIPS*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] c.f. Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental settings

A.1 Dataset statistics

We consider eight real-world graph datasets including citation networks Cora [32], Citeseer [32], Pubmed [23], Coauthor CS [34], Coauthor Physics [34], and CoraFull [2] together with Amazon co-purchase networks Computers [34] and Photo [34]. Table 4 summarizes their statistics.

Table 4: Benchmark dataset statistics.

	Cora	Citeseer	Pubmed	Computers	Photo	CS	Physics	CoraFull
Nodes	2,708	3,327	19,717	13,752	7,650	18,333	34,493	19,793
Edges	10,556	9,104	88,648	491,722	238,162	163,788	495,924	126,842
Features	1,433	3,703	500	767	745	6,805	8,415	8,710
Classes	7	6	3	10	8	15	5	70
Homophily	82.52%	70.62%	79.24%	78.53%	83.65%	83.20%	91.53%	58.61%

We report the homophily index proposed by Pei et al. [29], which provides a global view of the neighborhood similarity for a graph. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the homophily is defined as

$$\mathcal{H}(\mathcal{G}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \frac{\text{Number of node } i\text{'s neighbors who have the same label as } i}{\text{Number of } i\text{'s neighbors}}. \tag{11}$$

A.2 Details of model training setup

We follow the setting of Shchur et al. [34] to define GCN [14] and GAT [40] models. Both models consist of 2 layers and the hidden dimension is fixed to 64. For the multi-head layer in GAT, the number of attention heads is fixed to 8 with 8 hidden units per head. We implement the models and training pipelines in PyTorch [28] and PyTorch Geometric [5]. All models are trained for a maximum of 2000 epochs, using early stopping with a patience of 100 epochs. We choose Adam [13] as the optimizer with initial learning rate 0.01. We add a weight decay of 5e-4 for Cora, Citeseer, and Pubmed, and 0 for the rest.

We use stratified sampling to randomly select 15% of the nodes as observed set, mask out the output labels of the rest 85% of the nodes for test prediction, and ensure that the nodes with the same label are split proportionally. Following Kull et al. [19, 18], we further divide the labeled set with three-fold cross-validation. The bigger portions (10%) are used as training sets and the rest (5%) are used as validation sets. The GNN models (GCN and GAT) are trained on the training set, then used to predict the masked-out test set. Figure 8 illustrates the aforementioned data partition in our experiments. In total, we use 5 random data splits, three-fold cross-validation for each split, and 5 random model initializations per data partition, resulting in 75 total runs for each experiment.

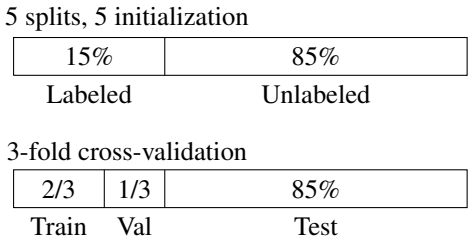

Figure 8: Data partition schema for graph datasets.

A.3 Details of model calibration setup

We compare GATS with four scaling-based calibrators: temperature scaling (TS) [7], vector scaling (VS) [7], ensemble temperature scaling (ETS) [43] , CaGCN [41]. The calibrators are trained on the validation set using the negative log-likelihood (NLL) loss and validated on the training set for

early stopping and hyperparameter search. The optimizer configuration and the training schedule are the same as Section A.2. We observe TS and VS using Adam with weight decay 0 achieves better performance than using L-BFGS [21] in the original implementation[3]. For ETS, we follow the official implementation that uses Sequential Least SQuares Programming (SLSQP) [15]. For CaGCN, we use a two-layer GCN with 16 hidden units and choose the hyperparameters following the original paper. For GATS, we utilize one message passing layer and initialize T_0, γ_t, and γ_n to 1 and ω to 0. We find the best hyperparameter using cross-validation. Table 5 shows the search space for GATS hyperparameters.

Table 5: Hyperparameter search space for GATS.

Hyperparameter	Search space
Weight decay	0, 1e-3, 5e-3, 1e-2, 5e-2, 1e-1, 2e-1, 3e-1
Initial T_0	1, 1.5

B Additional plots

Here we include additional plots which shows the corresponding factors influencing the calibration of GAT models (c.f. Section 4). Overall, we reach the same conclusions as the GCN case.

B.1 General under-confident tendency for GAT

Figure 9 summarizes the GAT results. We see a general tendency of under-confident predictions (plots above the diagonal) except for the Physics dataset, which differs from the overconfident behavior of multiclass image classification using CNNs.

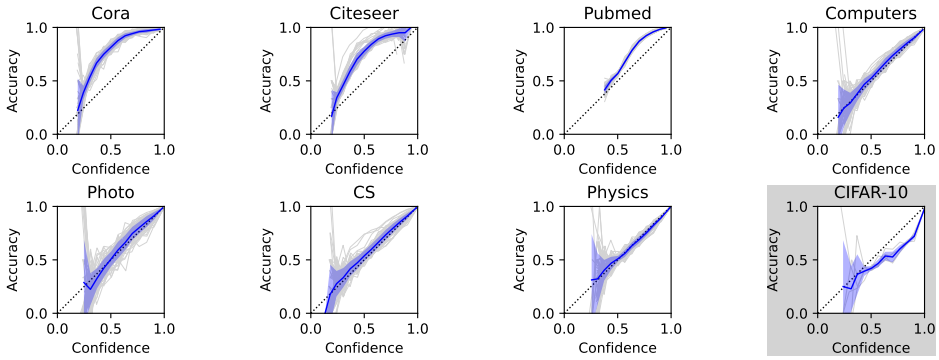

Figure 9: Reliability diagrams of GAT models trained on various graph datasets.

B.2 Diversity of node distributions for GAT

Figure 10 shows the GAT results. Compared to the standard classification case, predictions of GAT also tend to be more spread out.

[3] https://github.com/gpleiss/temperature_scaling

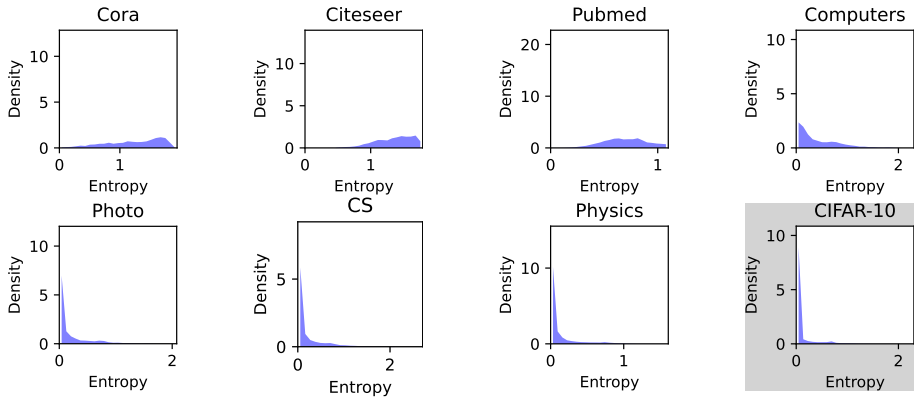

Figure 10: Entropy distributions of GAT predictions on graph datasets.

B.3 Effect of distance to training nodes for GAT

GAT result are shown in Figure 11. We also see that training nodes and their neighbors tend to be better calibrated.

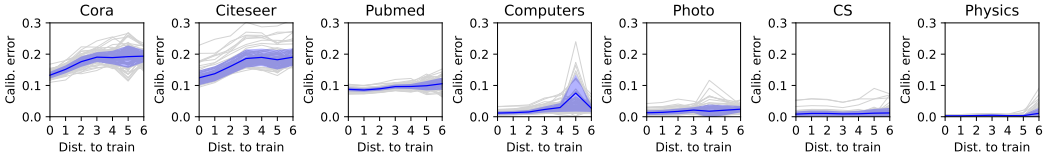

Figure 11: Nodewise calibration error of GAT results w.r.t the minimum distance to training nodes.

B.4 Relative confidence level for GAT

The plots for the GAT case are shown in Figure 12. Similar to the GCN case, we observe that nodes which are less confident than their neighbors tend to be poorly calibrated and it is in general desirable to have a comparable confidence level w.r.t. the neighbors.

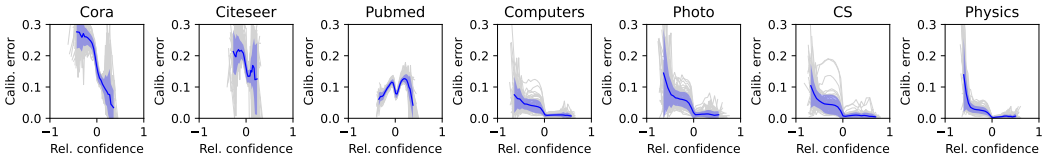

Figure 12: Nodewise calibration error of GAT results depending on the relative confidence level.

B.5 Neighborhood similarity for GAT

Figure 13 shows tha GAT results. Analogue to the GCN case, nodes with strongly agreeing neighbors tend to have lower calibration errors.

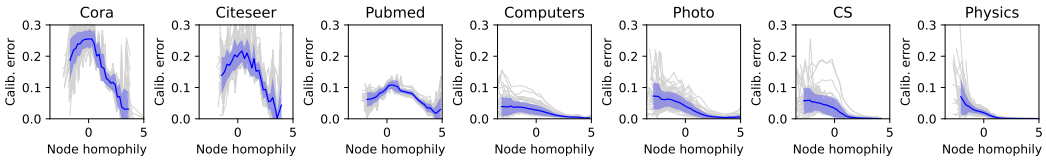

Figure 13: Nodewise calibration error of GCN results depending on the node homophily.

C Additional calibration results

This section includes supplementary results with additional metrics and calibration methods.

C.1 Results using additional metrics

Variants of ECE Even though ECE [27, 7] is the most commonly used metric for measuring calibration, it has some limitations: (1) ECE only considers top-1 probabilistic output but can not reflect classwise calibration. (2) The binning-based estimator [27] is dependent on the choice of binning scheme. To alleviate these disadvantages, we evaluate the trained calibrators using the classwise-ECE [19] and the kernel density estimation (KDE) based KDE-ECE [43] which is a binning-free metric:

- **Classwise-ECE** measures the gap between the classwise average prediction $\text{conf}(B_{m,k})$ and the actual frequency of that class $\text{freq}(B_{m,k})$ in M equally spaced bins across all classes $k \in K$:

$$\text{Classwise-ECE} = \frac{1}{K} \sum_{k}^{K} \sum_{m}^{M} \frac{|B_{m,k}|}{|\mathcal{N}|} |\text{freq}(B_{m,k}) - \text{conf}(B_{m,k})| \tag{12}$$

$$\text{freq}(B_{m,k}) = \frac{1}{|B_{m,k}|} \sum_{i \in B_{m,k}} \mathbf{1}(y_i = k) \tag{13}$$

$$\text{conf}(B_{m,k}) = \frac{1}{|B_{m,k}|} \sum_{i \in B_{m,k}} \hat{p}_{i,k} \tag{14}$$

  where $\hat{p}_{i,k}$ denotes the probability of predicting class k for sample i in bin $B_{m,k}$. We compute classwise-ECE using $M = 15$ bins in our implementation.

- Instead of using a binning-based estimator, **KDE-ECE** uses a kernel function K_h to estimate the accuracy $\tilde{\pi}(c)$ given confidence prediction c and the marginal density function $f(c)$ of the predictive confidence:

$$\text{KDE-ECE} = \int |\tilde{\pi}(c) - c| \, \tilde{f}(c) dc \tag{15}$$

$$\tilde{\pi}(c) = \frac{\sum_{i \in \mathcal{N}} \mathbf{1}(y_i = \hat{y}_i) K_h(c - \hat{c}_i)}{\sum_{i \in \mathcal{N}} K_h(c - \hat{c}_i)} \tag{16}$$

$$\tilde{f}(c) = \frac{h^{-1}}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} K_h(c - \hat{c}_i) \tag{17}$$

  where $i \in \mathcal{N} \subset \mathcal{V}$ denotes the evaluated node, and h is the bandwidth of the kernel function. We follow the official implementation[4] of KDE-ECE, where the the Triweight Kernel $K_h(u) = (1/h)\frac{35}{32}(1 - (u/h)^2)^3$ [3] on $[-1, 1]$ is chosen as the kernel function and bandwidth is calculated as $h = 1.06\sigma|\mathcal{N}|^{-1/5}$ [31] with σ being the standard deviation of the confidence.

The classwise-ECEs are summarized in Table 6, and the KDE-ECEs are collected in Table 7. In general, we observe similar conclusions as in the confidence ECE case (c.f. Section 6.1): Overall GATS achieves the state-of-the-art calibration results.

[4] `https://github.com/zhang64-llnl/Mix-n-Match-Calibration`

Table 6: GNN calibration results in terms of classwise-ECE (in percentage, lower is better).

Dataset	Model	Uncal	TS	VS	ETS	CaGCN	GATS
Cora	GCN	4.34±1.41	2.06±0.27	2.11±0.30	2.07±0.26	2.23±0.29	**2.03±0.24**
	GAT	7.24±0.46	2.35±0.23	**2.03±0.23**	2.34±0.24	2.24±0.26	2.34±0.28
Citeseer	GCN	4.51±1.86	2.85±0.40	2.77±0.39	2.82±0.42	3.16±0.47	**2.74±0.39**
	GAT	8.34±1.02	3.12±0.52	**2.86±0.48**	3.09±0.51	3.22±0.53	3.10±0.58
Pubmed	GCN	4.96±0.99	1.38±0.26	1.53±0.30	1.39±0.26	1.35±0.33	**1.26±0.26**
	GAT	8.54±0.49	1.94±0.31	1.96±0.27	1.94±0.31	**1.89±0.37**	2.00±0.32
Computers	GCN	0.97±0.16	0.93±0.11	0.91±0.12	0.95±0.11	**0.84±0.10**	0.89±0.08
	GAT	0.83±0.13	0.81±0.09	**0.80±0.10**	0.82±0.10	0.84±0.11	**0.80±0.08**
Photo	GCN	0.89±0.22	0.78±0.12	0.81±0.15	0.78±0.15	0.80±0.08	**0.76±0.10**
	GAT	0.92±0.26	0.84±0.15	**0.82±0.15**	0.84±0.17	0.89±0.12	0.83±0.15
CS	GCN	0.39±0.11	**0.29±0.02**	0.32±0.03	**0.29±0.02**	0.42±0.10	**0.29±0.03**
	GAT	0.39±0.15	0.34±0.04	0.34±0.03	0.34±0.04	0.47±0.09	**0.33±0.04**
Physics	GCN	0.39±0.11	0.36±0.06	**0.35±0.05**	0.36±0.06	0.47±0.15	0.36±0.05
	GAT	0.39±0.06	0.39±0.05	**0.37±0.04**	0.39±0.05	0.55±0.14	0.39±0.06
CoraFull	GCN	0.35±0.03	0.34±0.01	0.34±0.01	0.34±0.02	0.34±0.04	**0.33±0.02**
	GAT	0.33±0.03	0.32±0.01	0.32±0.02	0.32±0.01	0.34±0.06	**0.31±0.02**

Table 7: GNN calibration results in terms of KDE-ECE (in percentage, lower is better).

Dataset	Model	Uncal	TS	VS	ETS	CaGCN	GATS
Cora	GCN	13.35±5.07	3.21±1.18	3.56±1.25	3.26±1.23	4.17±1.48	**3.11±1.19**
	GAT	23.33±1.79	3.00±0.80	**2.79±0.81**	2.97±0.85	3.17±0.87	2.93±0.95
Citeseer	GCN	10.72±5.86	4.80±1.40	4.56±1.45	4.66±1.46	6.09±1.30	**4.17±1.31**
	GAT	22.86±3.54	4.42±1.46	3.79±1.53	4.32±1.42	5.17±1.23	**3.66±1.58**
Pubmed	GCN	7.33±1.48	1.32±0.27	1.57±0.38	1.35±0.29	1.29±0.48	**1.07±0.26**
	GAT	12.32±0.80	1.20±0.29	1.12±0.29	1.20±0.29	**1.08±0.30**	1.16±0.30
Computers	GCN	3.05±0.97	2.60±0.71	2.68±0.75	2.73±0.76	**1.65±0.62**	2.16±0.61
	GAT	1.89±1.00	1.62±0.63	1.70±0.69	1.67±0.72	1.75±0.62	**1.47±0.52**
Photo	GCN	2.59±1.29	1.82±0.87	1.94±0.92	1.90±0.95	**1.65±0.45**	1.67±0.70
	GAT	2.25±1.16	1.74±0.67	1.81±0.75	1.81±0.74	1.75±0.60	**1.73±0.65**
CS	GCN	2.14±0.98	1.10±0.11	1.11±0.17	1.09±0.11	1.95±0.90	**1.06±0.12**
	GAT	1.74±1.30	1.12±0.25	1.10±0.26	1.14±0.26	2.10±0.88	**1.07±0.21**
Physics	GCN	0.94±0.27	0.83±0.09	**0.82±0.07**	0.83±0.09	0.96±0.24	0.85±0.09
	GAT	0.84±0.10	0.84±0.08	0.85±0.09	0.84±0.08	1.07±0.24	**0.83±0.08**
CoraFull	GCN	6.46±1.30	5.45±0.43	5.66±0.41	5.43±0.45	5.73±2.54	**3.78±0.90**
	GAT	4.76±1.44	3.98±0.51	4.14±0.45	3.96±0.53	5.90±3.11	**3.56±0.66**

Non calibration metrics Although not calibration metrics, we also report the results in terms of negative log-likelihood (Table 8) and Brier score (Table 9) for reference.

C.2 Results for additional baselines

While in the main paper we focus on "temperature scaling style" methods which directly rescale the output logits, here we compare with the following additional calibration methods which have different principles. These methods are all designed for multi-class classification and do not consider the structural information of the graph.

- **Multi-class isotonic regression (IRM)** [43] is a multi-class generalization of the non-parametric isotonic regression method;

Table 8: GNN calibration results in terms of negative log-likelihood ($\times 10^{-2}$).

Dataset	Model	Uncal	TS	VS	ETS	CaGCN	GATS
Cora	GCN	62.90±5.68	56.37±3.12	57.66±4.37	56.01±3.00	66.88±7.78	55.91±3.17
	GAT	75.67±2.37	57.51±2.87	55.91±4.02	57.15±2.70	60.79±4.66	57.02±2.33
Citeseer	GCN	90.13±6.00	86.99±2.74	87.01±2.33	86.61±2.56	92.95±4.88	86.18±2.35
	GAT	100.90±6.33	86.57±3.30	86.01±1.87	86.18±2.79	89.07±3.40	86.20±3.22
Pubmed	GCN	39.31±1.47	36.75±0.68	36.79±0.69	36.53±0.67	35.97±1.16	36.39±0.62
	GAT	46.87±0.98	40.06±0.76	40.06±0.74	40.07±0.75	39.78±0.77	40.05±0.75
Computers	GCN	42.96±1.21	42.93±1.17	42.87±1.15	41.08±1.31	43.31±3.47	42.49±1.25
	GAT	37.26±1.53	37.18±1.48	37.07±1.38	36.54±1.60	40.38±4.06	37.11±1.54
Photo	GCN	28.92±1.20	29.02±1.18	29.25±1.31	27.19±1.24	37.59±7.72	28.81±1.23
	GAT	26.83±1.78	26.82±1.61	26.79±1.61	26.40±1.76	32.75±5.37	26.93±1.77
CS	GCN	21.85±0.74	21.38±0.48	21.65±0.45	21.36±0.46	27.38±5.57	21.28±0.49
	GAT	24.76±1.46	24.57±0.87	24.59±0.72	24.49±0.85	29.79±3.86	24.49±0.83
Physics	GCN	11.95±0.41	11.88±0.34	11.90±0.33	11.89±0.34	13.00±1.27	11.87±0.32
	GAT	12.88±0.41	12.88±0.39	12.84±0.38	12.88±0.39	13.52±0.67	12.87±0.39
CoraFull	GCN	143.07±2.02	142.71±1.80	142.85±1.98	141.74±1.61	146.55±12.81	140.10±1.92
	GAT	139.77±2.16	139.57±1.89	139.72±1.91	138.97±1.86	150.70±17.93	139.06±1.84

Table 9: GNN calibration results in terms of Brier score ($\times 10^{-2}$).

Dataset	Model	Uncal	TS	VS	ETS	CaGCN	GATS
Cora	GCN	28.68±2.54	25.62±0.98	25.65±1.05	25.62±0.97	26.19±1.05	25.59±1.07
	GAT	34.47±1.21	26.67±1.05	25.31±1.03	26.67±1.03	26.71±1.17	26.71±1.00
Citeseer	GCN	42.56±2.82	40.76±0.76	40.94±0.93	40.71±0.78	41.57±1.06	40.63±0.74
	GAT	47.42±3.26	40.62±0.90	40.44±0.62	40.55±0.90	40.99±0.97	40.61±0.95
Pubmed	GCN	21.31±0.71	20.20±0.36	20.24±0.38	20.20±0.36	20.05±0.41	20.17±0.36
	GAT	25.33±0.56	22.68±0.41	22.64±0.41	22.68±0.41	22.58±0.43	22.67±0.41
Computers	GCN	18.57±0.80	18.50±0.68	18.42±0.64	18.51±0.68	18.13±0.70	18.42±0.65
	GAT	16.79±0.80	16.76±0.75	16.61±0.64	16.76±0.75	16.84±0.73	16.75±0.73
Photo	GCN	11.72±0.66	11.60±0.59	11.62±0.65	11.60±0.60	11.67±0.51	11.56±0.56
	GAT	11.52±0.88	11.45±0.77	11.35±0.69	11.45±0.77	11.59±0.70	11.46±0.76
CS	GCN	10.28±0.27	10.16±0.20	10.20±0.19	10.16±0.20	10.60±0.42	10.14±0.21
	GAT	11.36±0.58	11.27±0.35	11.25±0.30	11.27±0.35	11.65±0.39	11.27±0.34
Physics	GCN	6.13±0.20	6.13±0.19	6.13±0.19	6.13±0.19	6.25±0.22	6.12±0.19
	GAT	6.54±0.18	6.54±0.18	6.53±0.17	6.54±0.18	6.64±0.18	6.53±0.18
CoraFull	GCN	52.32±0.68	52.09±0.51	52.01±0.49	52.08±0.51	52.20±1.20	51.61±0.54
	GAT	51.73±0.77	51.60±0.60	51.55±0.59	51.59±0.60	52.53±1.71	51.54±0.59

- **Calibration using spline (Spline)** [8] fits the calibration function with splines;
- **Dirichlet calibration (DIR)** [19] uses the Dirichlet distribution to model the distribution of probabilistic outputs. It also employs an off-diagonal and intercept regularization (ODIR);
- **Order invariant calibration (OI)** [30] is the order-invariant intra order-preserving model. It uses sorted output logits as calibration input and builds up a neural network with special structures to preserve the accuracy and the intra order of the predicted logits.

The authors of spline calibration specify how to calibrate a specific class or a chosen top-r class, and in their implementation[5] they focus on calibrating the top-1 class. However, it is not clear how to adjust the rest of the predictions to ensure valid probabilistic predictions after calibration. We adopt

[5] `https://github.com/kartikgupta-at-anu/spline-calibration`

a heuristic which proportionally rescales the non top-1 output probabilities so that the calibrated probabilistic output sums up to one. Also, the authors wrongly claimed that calibrating the top-1 score "does not alter the classification accuracy" [8]. In practice, the score after calibration might no longer remain top-1 and the predictions could be altered.

For Dirichlet calibration, we find out that the scaling factors λ, μ of ODIR affect the performance and need to be tuned depending on the dataset. Thus we do a hyperparameter search for each dataset with search space $(0.01, 0.1, 1, 10, 100, 1000, 10000, 100000)$.

Table 10 summarizes the calibration results in terms of ECE. Note that we do not include the results for Dirichlet calibration on CoraFull because it fails to calibrate the GNN backbones and significantly deteriorates the predictive accuracies ($< 10\%$ v.s. $> 60\%$ before calibration).

Overall we observe that GATS still achieves the state-of-the-art performance compared to the additional baselines for GNN calibration.

Table 10: GNN calibration results (ECE, in percentage) with additional baselines.

Dataset	Model	Uncal	IRM	Spline	DIR	OI	GATS
Cora	GCN	13.04±5.22	3.69±1.17	4.89±1.27	3.93±1.26	4.83±1.50	**3.64±1.34**
	GAT	23.31±1.81	3.45±0.91	4.71±1.76	3.42±0.72	4.24±1.39	**3.18±0.90**
Citeseer	GCN	10.66±5.92	5.08±1.34	6.70±1.42	5.40±1.52	6.36±1.48	**4.43±1.30**
	GAT	22.88±3.53	4.15±1.50	6.07±1.77	4.87±1.36	6.08±1.30	**3.86±1.56**
Pubmed	GCN	7.18±1.51	1.64±0.58	1.72±0.46	1.42±0.33	1.23±0.44	**0.98±0.30**
	GAT	12.32±0.80	1.63±0.60	1.69±0.60	**0.93±0.26**	1.36±0.47	1.03±0.32
Computers	GCN	3.00±0.80	1.98±0.48	**1.56±0.44**	3.31±0.63	1.86±0.55	2.23±0.49
	GAT	1.88±0.82	**1.32±0.35**	1.56±0.53	2.23±0.73	2.17±0.72	1.39±0.39
Photo	GCN	2.24±1.03	1.53±0.47	1.68±0.57	1.61±0.60	1.75±0.49	**1.51±0.52**
	GAT	2.02±1.11	1.53±0.51	1.59±0.66	1.39±0.62	1.85±0.66	**1.48±0.61**
CS	GCN	1.65±0.92	1.29±0.32	1.08±0.38	0.90±0.19	1.55±0.50	**0.88±0.30**
	GAT	1.40±1.25	1.09±0.35	1.16±0.39	0.96±0.39	1.80±0.80	**0.81±0.30**
Physics	GCN	0.52±0.29	0.59±0.17	0.54±0.23	**0.44±0.15**	0.64±0.29	0.46±0.16
	GAT	0.45±0.21	0.56±0.16	0.45±0.18	**0.42±0.14**	0.60±0.32	**0.42±0.14**
CoraFull	GCN	6.50±1.26	4.33±0.77	**2.92±0.79**	N/A	10.61±1.40	3.76±0.74
	GAT	4.73±1.39	3.18±0.56	**2.68±0.89**	N/A	8.33±2.18	3.54±0.63

C.3 Accuracies of calibration methods

Since many baseline calibration methods are not accuracy-preserving, in Table 11 we additionally report their test accuracies. Accuracy preserving methods (GATS, TS, ETS, CaGCN, OI) have the same accuracies as the uncalibrated case, which is also reported for reference.

D Data efficiency and expressiveness of GATS: GAT results

Figure 14 shows the results for the GAT case. We see that GATS is also data efficient and expressive when calibrating GAT models.

E CaGCN results discussion

While the ECEs of CaGCN in its original paper are promising [41], we observe that the ECEs of CaGCN are often unstable and sometimes even worse than that of the uncalibrated model in our experiments. One possible reason is that we use a different splitting from the CaGCN paper, where they follow a fixed splitting from Kipf and Welling [14]. A significant difference is that the splitting from Kipf and Welling [14] has more validation nodes than the training nodes. This differs from typical real-world applications, where the larger fold would often be used to train a good classifier [26], and only the smaller fold is available for fitting the calibrator.

Table 11: Test accuracies of uncalibrated results (identical to those from accuracy-preserving methods) and calibrated predictions from non accuracy-preserving methods.

Dataset	Model	Uncal	VS	IRM	Spline	DIR
Cora	GCN	82.78±0.79	82.90±0.89	82.56±0.87	82.78±0.80	83.16±0.87
	GAT	81.98±0.92	82.98±0.77	81.74±1.04	81.98±0.92	82.77±0.85
Citeseer	GCN	72.19±0.82	72.06±0.90	72.04±0.79	72.16±0.82	72.31±0.99
	GAT	72.37±0.68	72.25±0.64	72.18±0.73	72.34±0.73	72.53±0.59
Pubmed	GCN	86.40±0.27	86.39±0.29	86.31±0.29	86.39±0.26	86.43±0.25
	GAT	84.46±0.34	84.55±0.38	84.25±0.38	84.44±0.34	84.62±0.35
Computers	GCN	88.13±0.56	88.19±0.56	88.20±0.54	88.12±0.55	87.78±0.65
	GAT	89.05±0.60	89.16±0.52	89.03±0.60	89.04±0.60	89.00±0.63
Photo	GCN	92.65±0.38	92.69±0.43	92.61±0.42	92.65±0.38	92.69±0.49
	GAT	92.65±0.54	92.76±0.45	92.58±0.57	92.64±0.54	92.92±0.44
CS	GCN	93.33±0.15	93.29±0.15	93.29±0.16	93.32±0.15	93.33±0.15
	GAT	92.57±0.25	92.57±0.22	92.54±0.24	92.56±0.24	92.60±0.22
Physics	GCN	95.99±0.14	95.98±0.14	95.98±0.15	95.98±0.14	96.00±0.14
	GAT	95.70±0.13	95.71±0.12	95.67±0.14	95.69±0.14	95.72±0.11
CoraFull	GCN	63.07±0.50	63.24±0.45	62.94±0.48	63.07±0.50	N/A
	GAT	63.00±0.59	63.10±0.52	62.85±0.59	63.00±0.58	N/A

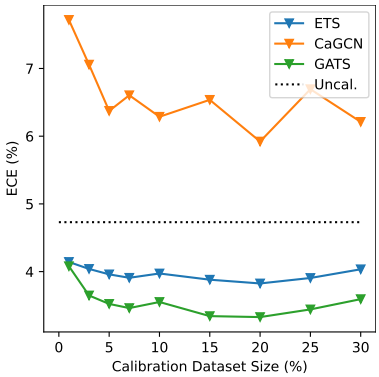

Figure 14: ECEs (in percentage) on CoraFull with GAT backbone for ETS, CaGCN, and GATS using various amounts of calibration data. Again, We observe that GATS is data-efficient and expressive. Here CaGCN fails to calibrate the GAT model on CoraFull.

In our splitting, the validation sets of Cora and Citeseer are substantially smaller than those in Kipf and Welling [14]. We observe that CaGCN yields suboptimal calibration results (see Section 6.1) and predictions with higher confidence tend to be over-confident in the reliability diagram in Figure 15. By contrast, the validation set in Pubmed is relatively large since it has more nodes. We notice that CaGCN achieves competitive results in Pubmed and the confidence-accuracy curve almost lies on the diagonal. We observe that CaGCN also produces suboptimal calibration results in CoraFull, even though the validation set is large. We suspect that this is caused by the class imbalance of the CoraFull data. Class imbalance is known to be a challenge for many calibration methods [36].

F GATS weight visualization

GATS learns τ_i^h from sorted logits. We discover that the absolute value of the learned weights in the linear layer generally follows the ranking of the logits across the class. That is to say, logits with higher value have stronger influence to τ_i^h. In Figure 16 we visualize the weights $\theta = (\theta^1, \ldots, \theta^H)$ of the linear layers ϕ^h in GATS. Here, it is interesting to see that the weights θ^h from different heads

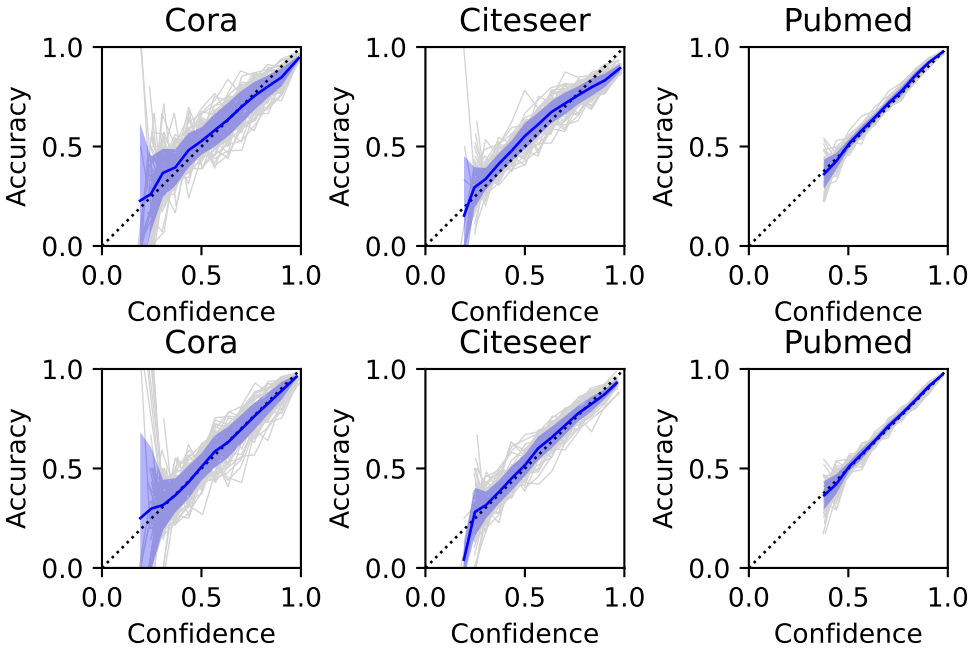

Figure 15: Top row: Reliability diagram of CaGCN for GCN trained on Cora, Citeseer, and Pubmed. Botom row: Reliability diagram of GATS for GCN trained on Cora, Citeseer, and Pubmed as a reference. We observe that CaGCN is noticeably over-confident in the high confidence region when the size of the validation set is relatively small (e.g., Cora and Citeseer).

h have slight variations. Combining multiple heads in the attention with sorting could be considered as a form of ensemble without the model being overly parameterized.

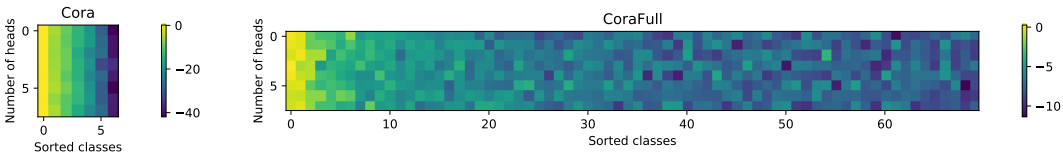

Figure 16: Visualization of the GATS weight θ on Cora (7 classes) and CoraFull (70 classes).

G Analysis of correlations between the factors

In this section we visualize the correlation between the local-view factors: the distance to training nodes, the relative confidence level, and the neighborhood similarity. Each plot shows how the factor on the y-axis varies when the factor on the x-axis is fixed to a given value. Two factors are independent when we observe a horizontal line in the plot. As the relative confidence level is a model dependent factor, GCN and GAT will have different correlation plots when it is involved.

G.1 Distance to training nodes – relative confidence level

Figures 17, 18, 19, and 20 show the correlation plots between the distance to training nodes and the relative confidence level. In Figure 17 and 19 we see that regardless of the distance to the training nodes, the averaged relative confidence level stays around zero.

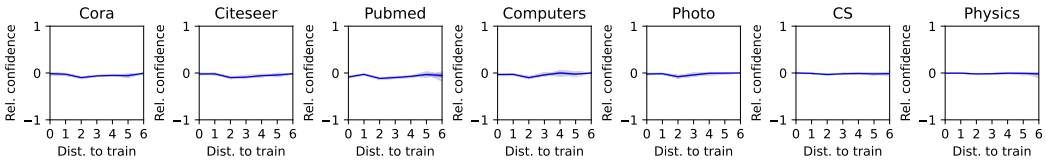

Figure 17: Relative confidence level of GCN results depending on the minimum distance to training nodes.

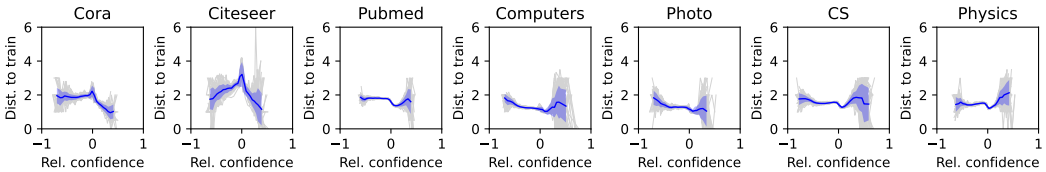

Figure 18: Minimum distance to training nodes depending on the relative confidence level of GCN predictions.

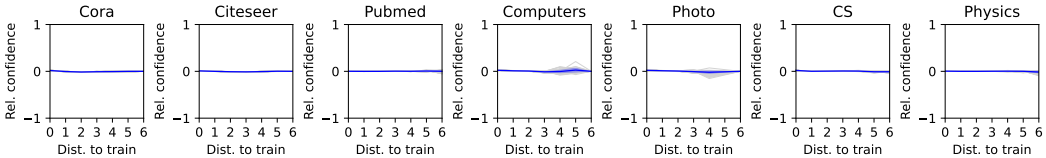

Figure 19: Relative confidence level of GAT results depending on the minimum distance to training nodes.

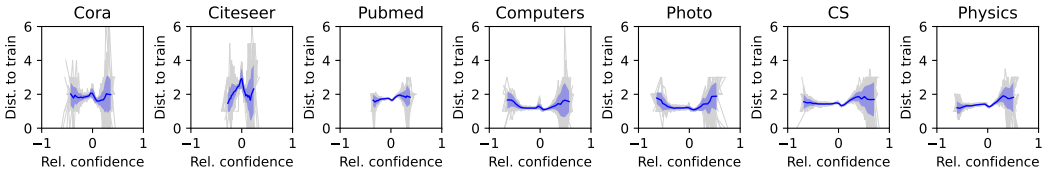

Figure 20: Minimum distance to training nodes depending on the relative confidence level of GAT predictions.

G.2 Relative confidence level – neighborhood similarity

Figures 21, 22, 23, and 24 show the correlations between the relative confidence level and the neighborhood similarity. We observe some partial correlation between these two factors, especially in the negative region of the node homophily.

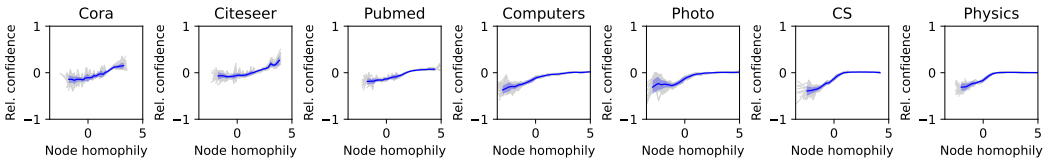

Figure 21: Relative confidence level of GCN predictions depending on the node homophily.

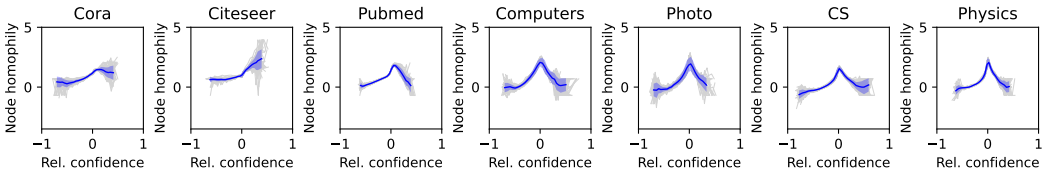

Figure 22: Node homophily depending on the relative confidence level of GCN predictions.

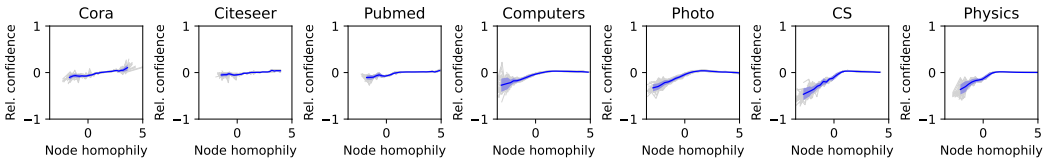

Figure 23: Relative confidence level of GAT predictions depending on the node homophily.

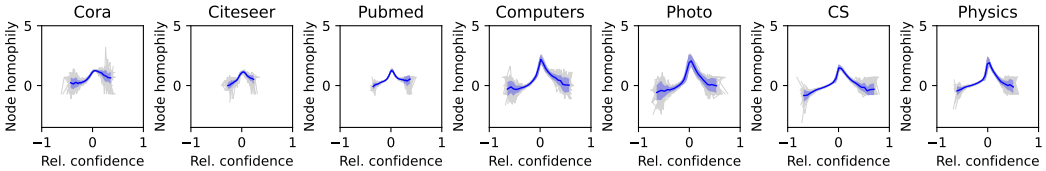

Figure 24: Node homophily depending on the relative confidence level of GAT predictions.

G.3 Distance to training nodes – neighborhood similarity

Figures 25 and 26 show the correlation between the distance to training nodes and neighborhood similarity. Note that these two factors are not model-dependent and thus GCN and GAT share the same results. We observe that these two factors have a less significant correlation.

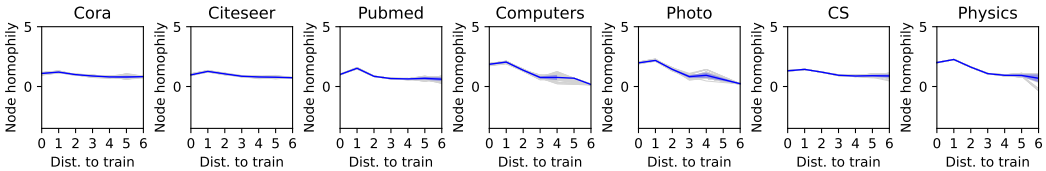

Figure 25: Node homophily depending on the minimum distance to training nodes.

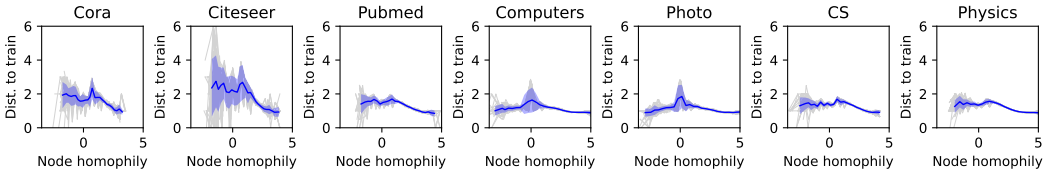

Figure 26: Minimum distance to training nodes depending on the node homophily.

H Factor node count analysis

In this section we plot the number of test nodes depending on the three local view factors: distance to training nodes, relative confidence level, and neighborhood similarity.

H.1 Node count for distance to training nodes

Figure 27 summarizes the node count results of the distance to training nodes. We see the majority of nodes can be connected to the training nodes by one or two hops.

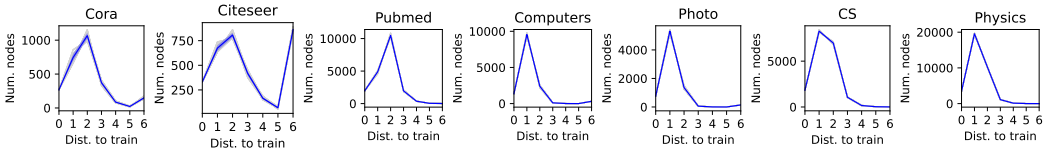

Figure 27: Number of nodes for the minimum distance to training nodes.

H.2 Node count for relative confidence level

Figure 28 and 29 are the node count results of the relative confidence level for GCN and GAT respectively. We observe that most of the nodes are concentrated around the zero relative confidence level.

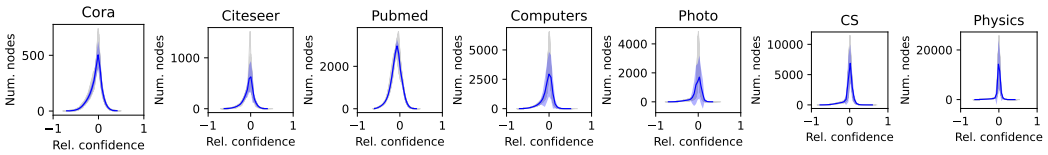

Figure 28: Number of nodes for the relative confidence level of GCN predictions.

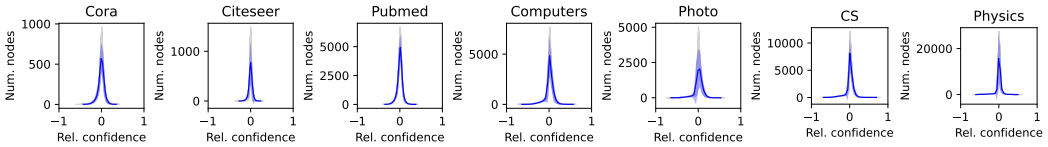

Figure 29: Number of nodes for the relative confidence level of GAT predictions.

H.3 Node count for neighborhood similarity

Figure 30 shows that the majority of nodes lie in the positive homophily region.

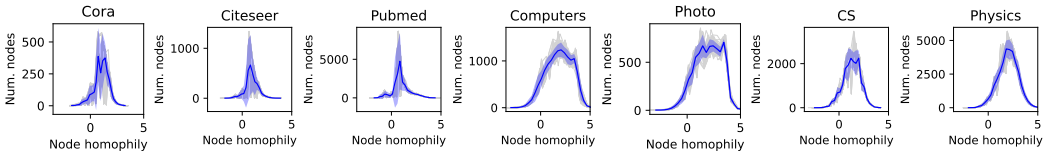

Figure 30: Number of nodes for the node homophily.