Understanding and Mitigating Under-Confidence in GNNs from the Final Layer

Jincheng Huang[1,2,4] **Jie Xu**[2] **Xiaoshuang Shi**[2] **Ping Hu**[2] **Lei Feng**[3*] **Xiaofeng Zhu**[1*]
[1]Hainan University [2]School of Computer Science and Engineering,
University of Electronic Science and Technology of China
[3]Southeast University [4]Singapore Management University

Abstract

Graph Neural Networks (GNNs) have demonstrated remarkable effectiveness on graph-based tasks. However, their predictive confidence is often miscalibrated, typically exhibiting *under-confidence*, which harms the reliability of their decisions. Existing calibration methods for GNNs normally introduce additional calibration components, which fail to capture the intrinsic relationship between the model and the prediction confidence, resulting in limited theoretical guarantees and increased computational overhead. To address this issue, we propose a simple yet efficient graph calibration method. We establish a unified theoretical framework revealing that model confidence is jointly governed by class-centroid-level and node-level calibration at the final layer. Based on this insight, we theoretically show that reducing the weight decay of the final-layer parameters alleviates GNN under-confidence by acting on the class-centroid level, while node-level calibration acts as a finer-grained complement to class-centroid-level calibration, which encourages each test node to be closer to its predicted class prototype in the final-layer representations. Extensive experiments validate the superiority of our method. The code is released at `https://github.com/huangJC0429/SCAR`.

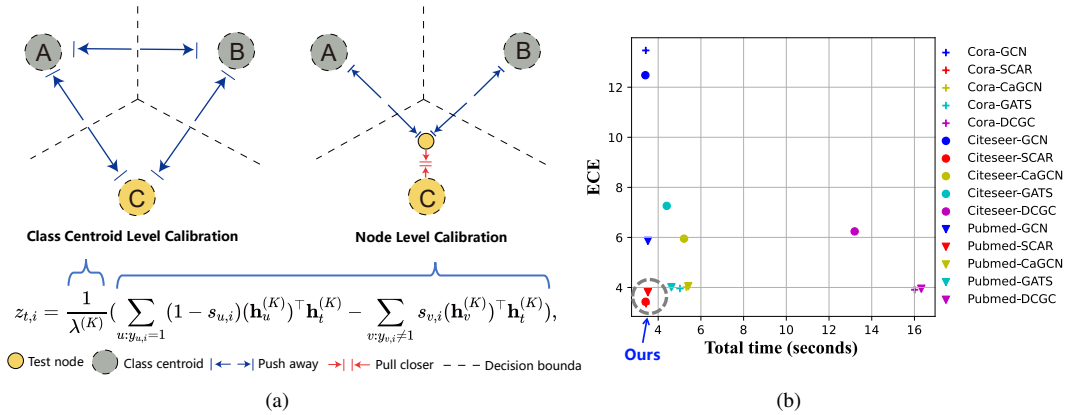
1 Introduction

Graph Neural Networks (GNNs) have achieved significant success in recent years, becoming a powerful tool for learning from graph-structured data. Their ability to effectively model complex relationships between nodes and edges has led to widespread applications, including molecular biology [5, 47], recommendation systems [8, 9], and knowledge graphs [55, 1]. Notably, even if the accuracy of GNNs meets high standards, the reliability of model outputs is critically important in real-world deployment. However, recent studies [46, 18, 44] have revealed that GNNs normally generate unreliable confidence, characterized by *under-confidence*—a sharp contrast to the over-confidence commonly observed in traditional Deep Neural Networks (DNNs). Consequently, traditional calibration methods cannot directly address the calibration issue of GNNs.

Current graph calibration methods can be roughly divided into two categories *i.e.,* regularization methods and post-hoc methods. Regularization methods [44, 40] introduce extra regularization terms specifically designed to calibrate GNNs during training. However, regularization methods were shown to struggle with effectively balancing accuracy and calibration [18, 50]. Post-hoc methods [46, 18, 41] are proposed to tune the confidence after model training, preserving the model's accuracy while improving the calibration performance, thereby overcoming the above issue. Most existing post-hoc calibration methods for GNNs adopt a common framework, where an additional calibration GNN is trained on the validation set to learn suitable temperature scaling coefficients, which are then applied via temperature scaling (TS) [16] for calibration. The main differences among these methods

*Corresponding authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) (b)

Figure 1: (a) An illustration of the proposed SCAR. The balls within the dotted box represent the class prototypes, while the yellow dots inside the solid box represent the test nodes classified as class C. The output logits are determined by two components. The first is the distance between class prototypes (*i.e.,* class-centroid-level calibration), which is controlled by the weight decay coefficient of the final layer. By reducing this coefficient, we push the class prototypes away from each other, increasing the class separability. The second component is the distance from each test node to its corresponding class prototype (*i.e.,* node-level calibration). By reducing this distance as a post-hoc method, we further refine the confidence of each test node in a more fine-grained manner to alleviate under-confidence. (b) Scatter plot illustrating the trade-off between runtime and ECE (lower is better). Scatters closer to the lower-left corner exhibit superior performance.

lie in the inputs to the calibration GNN. For example, CaGCN [46] takes the output probabilities of the base GNN; GATS [18] incorporates neighborhood similarity and predictive distributions, as well as other related factors; and SimCalib [41] leverages node similarity and homophily.

Despite the effectiveness of existing post-hoc graph calibration methods, there are still two limitations that need to be addressed. First, previous studies focus on alleviating the confidence bias through external operations , which lacks an in-depth exploration of the intrinsic relationship between the model itself and the under-confidence issue. Second, existing post-hoc calibration methods identify some potential factors (*i.e.,* the input of the calibration GNN) that are associated with model confidence, without uncovering how these factors affect it. Hence, they have to rely on additional neural networks and supervision from a validation set to learn the mapping in a black-box manner.

To address the above limitations, this paper conducts a theoretical analysis of the relationship between model parameter updates and prediction confidence, and proposes a novel method, **S**imple yet efficient graph **CA**lib**R**ation method (**SCAR**), as shown in Figure 1(a). To do this, we first reveal that the weight decay imposed on the final-layer parameters exacerbates the under-confidence issue of GNNs. To address this issue, we propose reducing the weight decay of the final-layer parameters to get a better-calibrated GNN. Moreover, our in-depth analysis shows that this adjustment is equivalent to enlarging the inter-class distance at the class-centroid level, thereby improving class separability. Since the confidence of each test node is closely related to its own representations, we introduce a node-level calibration strategy that encourages each test node to move closer to its predicted class prototype, serving as a fine-grained complement to class-centroid-level calibration. This method directly adjusts the representation of test nodes by reducing their distances to the prototypes, without learning additional mapping functions, thereby offering a training-free and interpretable solution. Finally, we establish a unified theoretical framework demonstrating that model confidence is jointly governed by both class-centroid-level and node-level calibration, highlighting the completeness and coherence of our approach, as shown at the bottom of Figure 1(a). Based on this, the proposed SCAR achieves state-of-the-art performance in both effectiveness and efficiency, as shown in Figure 1(b).

Compared with previous graph calibration methods[2], our contributions can be summarized as follows:

[2]The details of related work are summarized in Appendix A.

- To the best of our knowledge, we provide the first work to theoretically reveal that the weight decay imposed on the final-layer parameters exacerbates the under-confidence of GNNs by affecting the model at the class-centroid level. We address this issue by reducing the weight decay of the final-layer parameters.
- We propose a node-level calibration as a training-free post-hoc method, serving as a fine-grained complement to class-centroid-level calibration and offering a principled foundation for future post-hoc confidence calibration approaches.
- Theoretically demonstrate that model confidence is jointly governed by the proposed class-centroid-level and node-level calibration and empirically validate the effectiveness and efficiency of the proposed method across diverse settings, outperforming numerous state-of-the-art graph calibration methods.

2 Preliminary

Notations. Given a graph $\mathcal{G} = (V, E, \mathbf{X}, \mathbf{Y})$, where V is the node set and E is the edge set. The original node representation is denoted by the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ where n is the number of nodes and d is the number of features for each node. The label matrix is denoted by $\mathbf{Y} \in \mathbb{R}^{n \times c}$ with a total of c classes, and the training set is denoted as $\mathcal{T}$. The sparse matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of $\mathcal{G}$. Let $\mathbf{D} = \mathrm{diag}(d_1, d_2, \cdots, d_n)$ be the degree matrix, where $d_i = \sum_{j \in \mathcal{N}_i} a_{ij}$ is the degree of node i, the symmetric normalized adjacency matrix is represented as $\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-\frac{1}{2}}$ where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\mathbf{I}$ is the identity matrix and $\widetilde{\mathbf{D}}$ is the degree matrix of $\widetilde{\mathbf{A}}$.

Graph Neural Networks. Graph Neural Networks are powerful models for processing graph data. Most GNNs follow the message-passing framework. Specifically, every node iteratively updates its representations by aggregating the representations of its neighbors. Formally, the i-th layer of a GNN can be expressed as:

$$\mathbf{f}_v^{(i)} = \mathrm{COMB}(\mathbf{h}_v^{(i)}), \mathbf{h}_v^{(i)} = \mathrm{AGG}(\{\mathbf{f}_u^{(i-1)} : u \in \mathcal{N}_v\})), \tag{1}$$

where $\mathbf{f}_v^{(i)}$ is the representation of node v at the i-th layer, $\mathbf{h}_v^{(i)}$ is the representation after aggregation and $\mathcal{N}_v$ is a set of nodes adjacent to v. $\mathrm{AGG}(\cdot)$ denotes a differentiable, permutation-invariant function (e.g., sum, mean), $\mathrm{COMB}(\cdot)$ denotes a differentiable transformation function such as multi-layer perceptrons (MLPs). Different types of GNNs have different choices of $\mathrm{COMB}(\cdot)$ and $\mathrm{AGG}(\cdot)$. If there is no $\mathrm{AGG}(\cdot)$ in Eq. (1) (*i.e.,* $\mathbf{h}_v^{(i)} = \mathbf{f}_v^{(i-1)}$), then it becomes a traditional MLP.

Expected Calibration Error (ECE). Let $\mathbf{s}_v$ be the prediction probabilities of node v and the number of layers in the model is K, we have $\mathbf{s}_v = \mathrm{softmax}(\mathbf{z}_v)$, where $\mathbf{z}_v$ is the logits of node v. The confidence for node v is $p_v = \max_j s_{v,j}$. A model is said to be perfectly calibrated [46] if its confidence scores exactly correspond to the actual probabilities. For example, with 0.8 being the average confidence, 80% of the predicted examples should be correct. Nevertheless, GNNs are often found to exhibit *under-confidence*, where the confidence scores are lower than the actual probability. For example, more than 80% of predictions may be correct at an average confidence of 0.8.

$$\text{PC: } \Pr[\hat{y}_v = y_v | p_v = p] = p, \forall p \in [0, 1], \quad \text{UC: } \Pr[\hat{y}_v = y_v | p_v = p] > p, \forall p \in [0, 1], \tag{2}$$

where $\mathbf{PC}$ and $\mathbf{UC}$ denote perfect calibration and under-confidence, respectively. The calibration quality can be quantified by the expected calibration error (ECE) [32, 16]. It divides predictions into confidence intervals (bins, *i.e.,* $\mathrm{B}_1, \ldots, \mathrm{B}_M$) and calculates the accuracy and average predicted confidence within each bin. The ECE is then computed as the weighted average of the absolute differences between accuracy and confidence across all bins, with weights proportional to the number of predictions in each confidence bin. **A lower ECE means** that the confidence more closely matches its actual accuracy, thus indicating **better calibration.** Formally, the ECE can be defined as

$$\mathrm{ECE} = \sum\nolimits_{m=1}^{M} \frac{|B_m|}{|N_{\mathrm{test}}|} \left|\mathrm{acc}\left(B_m\right) - \mathrm{conf}\left(B_m\right)\right| \quad s.t. \begin{cases} \mathrm{acc}\left(B_m\right) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}\left(y_i = \hat{y}_i\right) \\ \mathrm{conf}\left(B_m\right) = \frac{1}{|B_m|} \sum_{i \in B_m} p_i \end{cases}, \tag{3}$$

where $|N_{\mathrm{test}}|$ is the number of test nodes and $\mathbf{1}(\cdot)$ is the indicator function.

3 The Proposed Method

Overview. To mitigate the under-confidence of GNNs by refining its inherent causes, we first theoretically show that the weight decay imposed on the final layer intensifies under-confidence. Thus, simply reducing the final layer's weight decay leads to better-calibrated predictions. Moreover, the in-depth analysis shows that this adjustment increases the distance between class prototypes by directly operating at the class-centroid-level. Given that a test node's confidence is also closely related to its own representation, we introduce a node-level calibration strategy as a fine-grained complement to class-centroid-level calibration. This strategy reduces the distance between each test node and its predicted class prototype in the final-layer representation space. Furthermore, our unified theoretical analysis reveals that model confidence is jointly governed by class-centroid-level and node-level calibration. An overview of the proposed method is presented in Figure 1(a).

3.1 Class-Centroid Level Calibration

Previous calibration methods calibrate the confidence through external methods. For example, GCL [44] designed a loss function that constrains the model outputs to have low entropy probabilities (*i.e.,* high confidence). CaGCN [46] trains a calibration model on the validation set to scale the output probabilities. However, they do not facilitate an in-depth exploration of the intrinsic relationship between the model itself and under-confidence. To address this issue, we analyze the impact of final-layer parameters on confidence during model updating, leveraging this to guide the model to produce better-calibrated confidence scores.

Specifically, the most commonly used cross-entropy loss with weight decay for node v is as follows:

$$\mathcal{L}_v = -\sum\nolimits_{i=1}^{c} y_{v,i} \log s_{v,i} + \sum\nolimits_{k}^{K} \frac{\lambda^{(k)}}{2} ||\mathbf{W}^{(k)}||_F^2, \tag{4}$$

where $\lambda^{(k)}$ is a regularization coefficient of layer k that controls the strength of the weight decay.

Then we have the following theorem (the proof is provided in the Appendix B.1):

Theorem 3.1. *Given the learning rate (i.e., η) and the final-layer parameters (i.e., $\mathbf{W}^{(K)}$). For an arbitrary node v in the training stage, its output probabilities on class i (i.e., $s_{v,i}$) is updated by:*

$$s'_{v,i} = \frac{e^{b_{v,i}/\tau}}{e^{b_{v,i}/\tau} + \sum_{j\neq i}^{c} e^{b_{v,j}/\tau} \cdot \psi_{i,j}}, \quad s.t. \begin{cases} \psi_{i,j} = e^{\eta(s_{v,i} - y_{v,i} - s_{v,j} + y_{v,j})(\mathbf{h}_v^{(K)})^\top \mathbf{h}_v'^{(K)}} \\ \tau = \frac{1}{1-\eta\lambda^{(K)}} \\ b_{v,i} = (\mathbf{W}_{:,i}^{(K)})^\top \mathbf{h}_v'^{(K)} \end{cases} \tag{5}$$

where $s'_{v,i}$ and $\mathbf{h}_v'^{(K)}$ are the GNN updated results of $s_{v,i}$ and $\mathbf{h}_v^{(K)}$ after the next epoch.

From Theorem 3.1, we can observe that during the update of output probabilities, the weight decay of the final layer only affects τ, and $\tau = \frac{1}{1-\eta\lambda^{(K)}} > 1$. Therefore, Theorem 3.1 reveals that applying weight decay to the final-layer parameters (*i.e.,* $\mathbf{W}^{(K)}$) is equivalent to adding a temperature coefficient greater than 1 to the output probability, which increases the entropy of output probabilities, thereby exacerbating the under-confidence of GNNs. Hence, reducing $\lambda^{(K)}$ can help mitigate the under-confidence issue specific to GNNs.

Although prior work [16] has observed that adjusting the global weight decay can influence model confidence, it often results in significant drops in accuracy, making it unsuitable as a practical calibration tool. To support this claim and highlight the strengths of our proposed method, we empirically compare the accuracy and ECE of global and final-layer weight decay in Figure 2. The experiment details can be found in section 4. The results show that regulating global weight decay is far less effective than our method in terms of calibration. More importantly, it negatively impacts accuracy, making confidence calibration meaningless, as effective calibration relies on a well-maintained accuracy level. Together, these results highlight that our work is the first to leverage weight decay as a practical calibration mechanism with theoretical support, while further indicating that its influence on confidence calibration essentially originates from the final layer.

Analyzing the effect of $\lambda^{(K)}$. Since reducing the final-layer weight decay (*i.e.,* $\lambda^{(K)}$) directly affects the parameter $\mathbf{W}^{(K)}$, this parameter plays a crucial role in generating confidence scores. To better

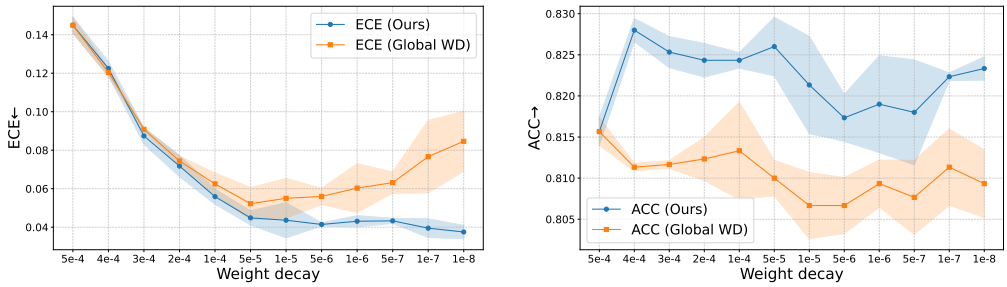


Figure 2: Impact of global vs. final-layer weight decay on Cora dataset (ECE(left), accuracy (right)).

understand its effect, we conduct an in-depth analysis of $\mathbf{W}^{(K)}$, which not only reveals that a smaller $\lambda^{(K)}$ increases the distance between class prototypes (*i.e.,* **class-centroid-level calibration**), but also highlights the necessity and lays the foundation for the design of subsequent node-level calibration. Specifically, we can analyze $\mathbf{W}^{(K)}$ by the following theorem (the proof is listed in Appendix B.2):

Theorem 3.2 (Optimality condition for $\mathbf{W}^{(K)}$)**.** *Given the objective function Eq. (4) with* $\mathbf{s}_v = \mathrm{softmax}((\mathbf{W}^{(K)})^\top \mathbf{h}_v^{(K)})$*, the optimizer* $\mathbf{W}^{(K)*}$ *satisfies the following stationary condition. For each class* $i \in \{1, \ldots, c\}$*,*

$$(\mathbf{W}_{:,i}^{(K)})^* = \frac{1}{\lambda^{(K)}} \Big(\sum\nolimits_{u \in \mathcal{T}: y_{u,i}=1} (1 - s_{u,i}^*) \mathbf{h}_u^{(K)} - \sum\nolimits_{v \in \mathcal{T}: y_{v,i} \neq 1} s_{v,i}^* \mathbf{h}_v^{(K)} \Big), \tag{6}$$

where s^* *denotes the softmax probability evaluated at the optimal solution* $\mathbf{W}^{(K)*}$*.*

From Theorem 3.2, we can see that each column of $(\mathbf{W}^{(K)})^*$ corresponds to a specific class prototype. For instance, $(\mathbf{W}_{:,i}^{(K)})^*$ corresponds to class i, where $i \in [1, \ldots, c]$. Moreover, $(\mathbf{W}_{:,i}^{(K)})^*$ is the weighted sum of the final-layer node representations belonging to class i in the training set, while subtracting the representations of nodes from other classes in the training set. Thus, every column of $(\mathbf{W}^{(K)})^*$ effectively represents the class prototypes of each class in the training set.

Building on the above observations, we can understand the effect of reducing the final-layer weight decay, according to the following proposition (the proof is provided in Appendix B.3):

Proposition 3.3. *Consider the final-layer optimization problem with fixed final-layer representations:* $\mathbf{W}_\lambda^* = \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \frac{\lambda}{2}\|\mathbf{W}\|_F^2$*, where* $\mathcal{L}(\mathbf{W})$ *is the cross-entropy loss and* λ *is the weight decay coefficient of the final-layer parameters. Let* $\mathbf{w}_i^*(\lambda)$ *denote the optimized weight vector of class* i*. For two coefficients* $\lambda_1 > \lambda_2 > 0$*, the average squared distance between final-layer class prototypes satisfies*

$$\frac{1}{c(c-1)} \sum_{i \neq j} \|\mathbf{w}_i^*(\lambda_1) - \mathbf{w}_j^*(\lambda_1)\|_2^2 \leq \frac{1}{c(c-1)} \sum_{i \neq j} \|\mathbf{w}_i^*(\lambda_2) - \mathbf{w}_j^*(\lambda_2)\|_2^2. \tag{7}$$

Therefore, reducing the final-layer weight decay enlarges the overall distance among final-layer class prototypes.

Proposition 3.3 shows that reducing the final-layer weight decay increases the Frobenius norm of the final-layer weight matrix. Since the centered final-layer weights of a softmax classifier can be interpreted as class prototypes, a larger Frobenius norm further implies a larger average pairwise distance among these prototypes. Therefore, reducing the final-layer weight decay improves class-centroid-level separability, which provides a theoretical explanation for why it alleviates the under-confidence issue of GNNs.

3.2 Node-Level Calibration

The above method performs calibration at the class-centroid level by controlling the distances between class prototypes. Since the confidence of each test node is also closely related to its own representation, it is necessary to adjust each node's representation as a fine-grained complement to the

class-centroid-level calibration, enabling more precise control over individual prediction confidence. Fortunately, based on the above theoretical analysis, we can easily derive the relationship between a test node's representation and its confidence. Specifically, given the optimality condition of the final-layer parameters (*i.e.,* $(\mathbf{W}^{(K)})^*$) in Eq. (6), we can calculate the output logit for arbitrary test node t:

$$z_{t,i} = ((\mathbf{W}_{:,i}^{(K)})^*)^\top \mathbf{h}_t^{(K)} = \frac{1}{\lambda^{(K)}} (\underbrace{\sum_{u:y_{u,i}=1} (1 - s_{u,i}^*)(\mathbf{h}_u^{(K)})^\top \mathbf{h}_t^{(K)}}_{\text{intra-class similarity}} - \underbrace{\sum_{v:y_{v,i}\neq 1} s_{v,i}^* (\mathbf{h}_v^{(K)})^\top \mathbf{h}_t^{(K)}}_{\text{inter-class similarity}}), \tag{8}$$

where $\mathbf{h}_t^{(K)}$ is the final-layer aggregated representation of $\mathbf{x}_t$. In Eq. (8), the first factor $\frac{1}{\lambda^{(K)}}$ has already been analyzed in the previous section. Therefore, we now focus on the second factor. This factor reveals that for an arbitrary test node t, the output logits for class i (*i.e.,* $z_{t,i}$) are composed of two terms (*i.e.,* intra-class similarity and inter-class similarity). The first term calculates the similarity between the i-th class prototype (*i.e.,* $\sum_{u:y_{u,i}=1}(1 - s_{u,i})(\mathbf{h}_u^{(K)})^\top$) and the test node t (*i.e.,* $\mathbf{h}_t^{(K)}$). The second term subtracts the similarity values between the test node t and the other class prototypes, excluding the prototype of class i (*i.e.,* $\sum_{v:y_{v,i}\neq 1} s_{v,i}(\mathbf{h}_v^{(K)})^\top$). Then we can conclude that in the final layer if the representation of a test node t is closer to the i-th class prototype and farther from other class prototypes, the i-th element of the logits (*i.e.,* $z_{t,i}$) will obtain a large positive value; otherwise, it will obtain a large negative value. Obviously, when the logit corresponding to the predicted class is a very large positive value and the logits for the other classes are very large negative values, the softmax function produces a high-confidence prediction. This inspires us that **we can increase the confidence of GNN by shortening the distance between the test node representation and the corresponding classified class prototype of the training set in the final layer.**

Based on the above analysis, we design node-level calibration, a **post-hoc** and **training-free** method. Its schematic is provided in Appendix C.1. After training the GNN is completed, we can adjust the final-layer representation of the test node t (*i.e.,* $\mathbf{h}_t^{(K)}$) using the following equation:

$$\mathbf{h}_t^{(K)} = \alpha \mathbf{W}_{:,i}^{(K)} + (1 - \alpha)\mathbf{h}_t^{(K)}, \;\; s.t.\; \hat{\mathbf{y}}_{t,i} = 1 \tag{9}$$

where $\alpha \in [0, 1]$ is a hyperparameter to control the similarity between node representation and its predicted class prototype, and $\hat{y}_t$ is the predicted label of node t. As we analyzed above, each column of $\mathbf{W}^{(K)}$ represents each class prototype. Therefore, Eq. (9) encourages each test node to move closer to its predicted class prototype. Meanwhile, the class-centroid-level calibration increases the distance between class prototypes, which in turn pushes the test node t farther away from other class prototypes. This is fully consistent with the analysis of Eq. (8).

Remark. Eq. (8) provides a unified formulation that reveals model confidence is jointly determined by two multiplicative factors. As shown in Figure 1(a), the first factor corresponds to class-centroid-level calibration, which enhances global class separability; the second factor corresponds to node-level calibration, which adjusts each node's representation relative to its predicted class prototype. This factorized form reflects the complementary nature of the two calibration strategies and highlights the completeness of our proposed method.

Analyzing GNN Bias in Node-Level Calibration. In the proposed node-level calibration, the class-centroid (*i.e.,* $\mathbf{W}^{(K)}$) is calculated by the training nodes. Therefore, the distance between the test node and the training node is a key factor for the node-level calibration. However, within the GNN framework, the distance from the test node to the training node is biased. Specifically, test nodes that are closer to training nodes in graph structure tend to obtain more similar representations to them through the GNN's message-passing mechanism [18, 4, 38]. We formalize this effect theoretically (The complete proof is provided in Appendix B.4.)

Proposition 3.4 (Effect of Message Passing)**.** *Consider a K-layer linearized GNN $\mathbf{H}^{(K)} = \mathbf{P}^K \mathbf{X}\mathbf{\Theta}$ with a nonnegative row-normalized propagation matrix $\mathbf{P}$, where $P_{vu} > 0$ only if $u \in \mathcal{N}_v \cup \{v\}$. Let $\mathbf{w}_i = \mathbf{W}_{:,i}^{(K)}$ be the prototype of class i, $\mathcal{T}_i$ the training nodes of class i, and $\rho_{v,i}^{(K)} = \sum_{u\in\mathcal{T}_i} [\mathbf{P}^K]_{vu}$ the same-class training influence on node v. Then,*

$$\|\mathbf{h}_v^{(K)} - \mathbf{w}_i\|_2 \leq R_i - \rho_{v,i}^{(K)}(R_i - \delta_i),$$

where $\delta_i < R_i$ denote the corresponding prototype-deviation bounds. Hence, weaker same-class training influence leads to a larger upper bound on the prototype deviation. Moreover, $d_{\mathcal{G}}(v, \mathcal{T}_i) > K$ implies $\rho_{v,i}^{(K)} = 0$.

Proposition 3.4 shows that test nodes receiving weaker same-class training influence, *e.g.*, those more than K hops away from the training nodes of their class, tend to deviate more from the corresponding class prototype and therefore motivate a stronger correction. Based on this observation, we further refine the proposed node-level calibration to mitigate the bias:

$$\begin{cases} \mathbf{h}_f^{(K)} = \alpha \mathbf{W}_{:,i}^{(K)} + (1-\alpha)\mathbf{h}_f^{(K)}, \;\; s.t.\; \hat{\mathbf{y}}_{f,i} = 1, f \in V_F \\ \mathbf{h}_s^{(K)} = \beta \mathbf{W}_{:,i}^{(K)} + (1-\beta)\mathbf{h}_s^{(K)}, \;\; s.t.\; \hat{\mathbf{y}}_{s,i} = 1, s \in V_S \end{cases}, \tag{10}$$

where V_F denotes the first-order neighbors of the training node, and V_S denotes its second-order and higher-order neighbors. The hyperparameters $\alpha, \beta \in [0, 1],\ \beta > \alpha$ control the similarity between node representation and class prototype, and $\hat{y}_v$ is the predicted label of node v.

In Eq. (10), we implement two levels of instance-wise tuning. First, the representation of every test node will be close to the centroid of its predicted class and pushed away from other classes' prototypes. Second, nodes that are not adjacent to the training set will receive greater weights to move closer to their corresponding class prototypes compared to nodes that are adjacent to the training set.

4 Experiments

In this section, we conduct experiments on four public datasets to evaluate the proposed method in terms of different settings. Details of experiments are shown in Appendix D, and additional experiments are shown in Appendix E.

4.1 Experimental Setup

Datasets. Following [46], we evaluate on four commonly used citation networks: Cora [39], Citeseer [39], Pubmed [39], and CoraFull [3]. Results on more diverse datasets are provided in Appendix E.3.

GNNs to be calibrated. In our experiments, we use the two most classic GNNs (*i.e.,* GCN [27] and GAT [42]) as backbones to be calibrated. For GCN and GAT, we follow parameters suggested by [27] and [42]. Additional results on more base models are presented in Appendix 4.4.

Comparison Methods. The comparison methods include two commonly used post-hoc calibration methods in general neural networks (*i.e.,* TS [16] and MS [28]), one regularization method specially designed for GNNs (*i.e.,* AU-LS [45]), and three SOTA post-hoc methods specially designed for GNNs (*i.e.,* CaGCN [46], GATS [18], DCGC [50], and GETS [60]).

Evaluation Protocol. We follow the evaluation in previous works [46], adopting Expected Calibration Error (ECE) with 20 bins as the metric for calibration. Besides, since fine-tuning the hyper-parameter of the weight decay in the final layer, the predicted label may have slight changes, we also adopt classification accuracy as an evaluation metric. We report the average and standard deviation of 10 runs for each pair split of a dataset.

4.2 Effectiveness Analysis

We first evaluate the effectiveness of the proposed method on the four citation datasets under different label rates. We are reporting the calibration results evaluated by ECE in Table 1 and Table 2. Obviously, our method achieves the best performance in calibration tasks. More surprisingly, the overall accuracy of our method has also improved. The specific results are in Appendix E.2. In addition, experiments on heterophilic and large-scale graphs are provided in Appendix E.3.

First, compared with the traditional calibration methods (*i.e.,* TS and MS), the proposed SCAR always outperforms them by large margins. For example, the proposed SCAR on average improves by 45.6%, compared to the best traditional calibration method (*i.e.,* TS, MS behaves badly on datasets with many classes, e.g., CoraFull) based on GCN and GAT. This demonstrates that calibration on graphs is more difficult and requires specialized methods.

Table 1: ECE (%) with 20 bins on different models for citation networks, considering various numbers of labels per class (L/C). "-" denotes the result is not meaningful. *Uncal.* represents the uncalibrated model. The best results are highlighted in black.

Datasets	L/C	GCN								
		Uncal.	**TS**	**MS**	**CaGCN**	**GATS**	**AU-LS**	**DCGC**	**GETS**	**SCAR**
Cora	20	13.47±0.63	4.88±0.55	4.14±0.57	4.01±0.67	3.96±0.46	4.32±0.23	3.91±0.36	3.83±0.41	**3.35±0.65**
	40	11.34±0.47	4.17±0.72	3.72±0.46	4.07±0.54	3.82±0.74	3.69±0.38	3.53±0.65	3.72±0.39	**2.67±0.33**
	60	9.37±0.49	3.55±0.54	3.64±0.61	3.74±0.44	3.56±0.36	4.09±0.81	3.42±0.66	3.26±0.67	**2.60±0.62**
Citeseer	20	12.48±0.71	6.41±0.87	6.44±0.37	5.95±0.72	7.26±0.63	6.69±1.76	6.24±0.41	6.72±0.78	**3.43±0.58**
	40	9.57±0.77	6.01±0.42	5.38±0.57	5.45±0.55	6.82±0.25	7.56±2.19	6.16±0.52	6.01±0.63	**4.17±0.45**
	60	8.06±0.64	5.59±0.50	5.21±0.64	5.46±0.34	8.11±0.37	5.74±0.99	7.43±0.61	7.84±0.66	**4.72±0.69**
Pubmed	20	5.86±0.77	5.41±0.38	4.76±0.42	4.05±0.60	4.01±0.13	6.48±1.35	3.95±0.21	3.93±0.37	**3.81±0.53**
	40	4.44±0.55	4.46±0.63	4.36±0.63	4.02±0.40	3.89±0.52	5.16±1.25	3.65±0.44	3.72±0.29	**3.16±0.52**
	60	4.45±0.97	3.67±0.60	3.18±0.64	3.11±0.48	3.23±0.43	5.09±0.68	3.13±0.14	3.03±0.26	**2.67±0.63**
CoraFull	20	19.86±0.61	10.31±0.61	-	7.76±0.64	7.39±0.75	7.37±0.89	7.36±0.84	7.01 ±0.85	**6.96±0.48**
	40	23.21±0.54	11.17±0.65	-	7.01±0.39	6.97±0.40	5.72±0.67	6.84±0.52	5.98±0.74	**5.61±0.43**
	60	23.37±0.40	9.81±0.38	-	7.68±0.34	6.65±1.67	7.28±0.96	6.59±1.01	6.83±0.41	**6.31±0.69**

Table 2: ECE (%) with 20 bins on different models for citation networks, considering various numbers of labels per class (L/C). "-" denotes the result is not meaningful. *Uncal.* represents the uncalibrated model. The best results are highlighted in black.

Datasets	L/C	GAT								
		Uncal.	**TS**	**MS**	**CaGCN**	**GATS**	**AU-LS**	**DCGC**	**GETS**	**SCAR**
Cora	20	15.58±0.89	7.17±0.98	5.44±0.94	4.50±0.56	4.25±0.29	5.51±0.83	4.10±0.53	4.53±0.48	**3.52±0.74**
	40	13.40±0.54	4.85±0.77	4.91±0.60	3.65±0.56	3.68±0.67	3.79±0.58	4.28±0.42	4.01±0.68	**3.52±0.73**
	60	12.01±0.33	3.93±0.61	4.11±0.53	3.13±0.32	2.76±0.31	3.44±0.95	2.64±0.63	3.16±0.58	**2.59±0.43**
Citeseer	20	15.34±0.50	9.16±0.87	6.33±0.98	5.72±0.68	6.01±0.65	8.12±0.13	5.88±0.15	5.85±0.33	**4.37±0.83**
	40	12.52±0.87	7.97±0.31	5.90±0.54	5.32±0.54	6.19±0.85	7.24±0.87	5.26±0.48	5.43±0.47	**3.54±0.65**
	60	10.90±0.59	6.48±0.71	5.19±0.91	5.25±0.76	5.48±0.53	6.47±0.60	5.44±0.36	5.39±0.76	**4.24±0.35**
Pubmed	20	8.35±0.31	6.56±0.46	5.01±0.37	3.56±0.63	4.68±1.68	3.58±0.47	**3.52±0.21**	4.16±0.37	3.78±0.84
	40	8.69±0.46	6.58±0.65	5.39±0.60	3.08±0.54	3.60±0.26	4.26±0.81	3.54±0.58	3.34 ±0.44	**3.02±0.30**
	60	9.93±0.41	6.69±0.63	5.39±0.60	3.08±0.54	2.94±0.72	3.97±0.52	2.90±0.98	2.96±0.61	**2.80±0.85**
CoraFull	20	21.19±0.36	11.01±0.51	-	7.88±0.60	6.94±0.53	8.89±0.59	6.64±0.63	6.79±0.71	**6.41±0.63**
	40	24.38±0.42	11.33±0.83	-	7.38±0.60	6.19±0.45	8.15±0.67	5.84±0.52	5.95±0.54	**5.52±0.46**
	60	24.97±0.18	11.33±0.52	-	8.49±0.69	5.58±0.66	8.36±0.63	5.34±0.63	5.21±0.69	**4.75±0.64**

Second, compared to graph calibration methods, the proposed SCAR achieves the best results, followed by CaGCN, GATS, AU-LS, DCGC, and GETS. For example, our method on average improves by 16.9%, compared to the most recent graph calibration method GETS, across all datasets and label rates, whether the base model is GCN or GAT. This can be attributed to the fact that the proposed SCAR calibrates the confidence of GNNs from the model itself, enabling the calibration process to better align with the model's inherent representation and learning dynamics.

4.3 Ablation Study

The proposed SCAR consists of two components: class-centroid-level calibration (**+ CLC** for short) and node-level calibration (**+ NLC** for short). To verify the effectiveness of each component in the proposed method, we evaluate the ECE of all variants using GCN and GAT backbones across all datasets with L/C = 20. The results are reported in Table 3.

According to Table 3, we have the following conclusions. First, our method with the complete components achieves the best performance. For example, our method on average improves by 7.9%, compared to the best variant (*i.e.,* **+ NLC**), indicating that all the components are necessary for our method. This is consistent with our analysis in Section 3. Those are the two proposed components that promote each other and complement each other. Second, our method significantly outperforms traditional TS in terms of any individual component's performance. For example, the individual **CLC** and **NLC** on average improve by 24.3% and 25.1% compared with TS. This indicates that the proposed method not only enhances the overall performance but also strengthens the contribution of each component. More importantly, previous post-hoc graph calibration methods were based on TS and needed to employ an extra calibration model to learn the appropriate temperature coefficients for

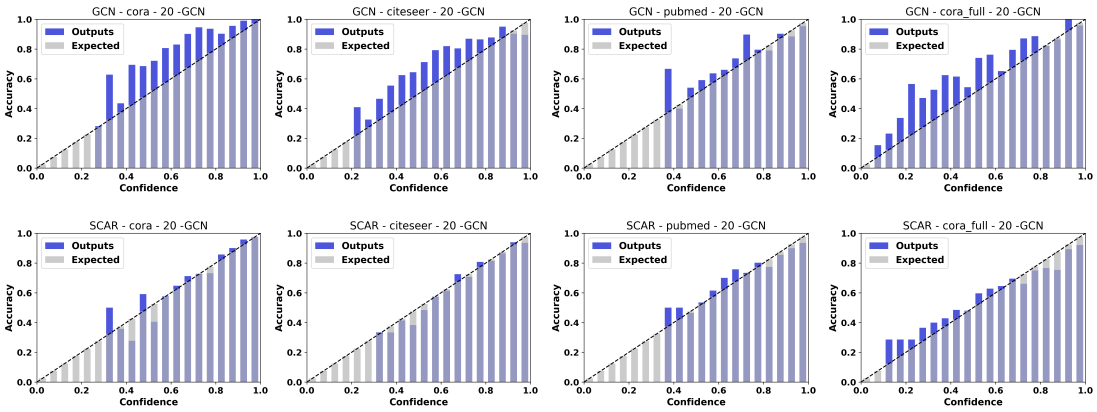


Figure 3: Visualization of Reliability diagrams on Cora, Citeseer, Pubmed, and CoraFull datasets.

every test node. The proposed NLC, like TS, is also a training-free post-hoc method, but its superior performance highlights its potential as a stronger foundation for subsequent post-hoc approaches.

Table 3: ECE (%) of each component with L/C=20. The best results are highlighted in black.

+CLC	+NLC	GCN				GAT			
		Cora	**Citeseer**	**Pubmed**	**CoraFull**	**Cora**	**Citeseer**	**Pubmed**	**CoraFull**
TS		4.88±0.55	6.41±0.87	5.41±0.38	10.13±0.61	7.17±0.98	9.16±0.87	6.56±0.46	11.01±0.51
✓		3.72±0.53	4.92±0.79	4.01±0.47	7.09±0.52	3.64±0.63	5.98±0.62	3.99±0.53	6.89±0.83
	✓	4.08±0.62	3.63±0.63	3.88±0.44	9.23±0.55	4.18±0.72	4.96±0.66	4.13±0.42	7.12±0.45
✓	✓	**3.35±0.65**	**3.43±0.58**	**3.81±0.53**	**6.96±0.48**	**3.52±0.74**	**4.37±0.83**	**3.78±0.84**	**6.41±0.63**

4.4 Validation on More Base Models

Table 4: ECE (%) with 20 bins on different models for different base models (*i.e.,* GCNII and FAGCN). The best results are highlighted in black.

ECE	Cora	Citeseer	Pubmed	Chameleon	Squirrel
GraphSAGE	9.89±0.78	10.08±0.88	7.38±0.65	13.27±0.81	11.14±0.48
SCAR-GraphSAGE	**4.88±0.71**	**4.32±0.73**	**5.86±0.46**	**6.74±0.86**	**7.17±0.51**
FAGCN	15.94±0.17	20.56±4.5	5.83±0.63	15.83±0.63	10.36±0.24
SCAR-FAGCN	**4.57±0.60**	**4.92±1.4**	**3.74±0.71**	**6.96±0.84**	**7.85±0.43**
GCNII	35.51±0.56	35.30±1.11	5.72±1.77	18.13±0.26	17.79±0.61
SCAR-GCNII	**9.81±0.48**	**8.74±0.51**	**3.98±0.88**	**7.03±0.35**	**8.93±0.37**

Beyond GCN and GAT, which are the most commonly used backbones in semi-supervised node classification, we further evaluate SCAR on more powerful architectures, including inductive GNN (*i.e.,* GraphSAGE [17]), heterophilic GNN (i.e., FAGCN [2]), and DeepGNN (i.e., GCNII [6]). These models incorporate advanced mechanisms such as residual connections and adaptive filtering, enabling them to capture more complex patterns in graph data. We conduct experiments on five datasets: Cora, Citeseer, and Pubmed (with 20 labeled nodes per class), as well as Chameleon and Squirrel, which are known to be heterophilous. The results are summarized in the Table 4.

The results of the Table 4 demonstrate that SCAR consistently improves calibration (ECE) across these models. This is because our method does not rely on any assumptions regarding graph homophily or specific message-passing mechanisms.

4.5 Visualization Analysis

In order to provide an intuitive and clear understanding of the effectiveness of the proposed model. We present visualizations of reliability diagrams L/C = 20, across different datasets.

In Figure 3, the first row represents the Reliability diagrams of GCN, and the second row represents the Reliability diagrams of the proposed SCAR. The x-axis divides the model's confidence into 20 equal intervals, while the y-axis represents the average accuracy for each interval. The gray area indicates the expected output values and the blue area represents the model's actual output values. The closer the bars are to the diagonal line (*i.e.,* $y = x$), the better calibrated the model. In all datasets, the uncalibrated GCN shows higher average accuracy than confidence in most bins, indicating under-confidence. Our proposed method effectively mitigates this under-confidence issue, aligning the model's confidence more closely with its accuracy.

5 Conclusion

In this paper, we conduct a comprehensive analysis of confidence calibration in GNNs. To do this, we first theoretically reveal that the weight decay imposed on the final-layer parameters aggravates the under-confidence of GNNs by shrinking the class prototypes toward the origin, which reduces class separability. To mitigate this issue, we propose simply reducing the final-layer weight decay to enhance class-centroid distinction and improve confidence calibration at the class-centroid level. Meanwhile, the node-level calibration strategy is proposed as a fine-grained complement to class-centroid-level calibration, which encourages each test node to be closer to its predicted class prototype and far away from other class prototypes in the final-layer representation space, thereby enhancing individual confidence calibration. Finally, we establish a unified theoretical framework showing that model confidence is jointly governed by both class-centroid-level and node-level calibration, which highlights the completeness and coherence of our method. Overall, the proposed method consistently outperforms state-of-the-art methods in both effectiveness and efficiency.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2022YFA1004100, the Key Program of the National Natural Science Foundation of China under Grant No. 62636004, the Hainan Provincial Key Research and Development Program under Grant No. ZDYF2026GXJS020, the National Natural Science Foundation of China under Grant Nos. 62476048 and 62276052, and the Sichuan Science and Technology Program under Grant No. 2025YFMS0004.

References

[1] Siddhant Arora. A survey on graph neural networks for knowledge graph completion. *arXiv preprint arXiv:2007.12374*, 2020.

[2] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *AAAI*, volume 35, pages 3950–3957, 2021.

[3] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[4] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

[5] Hanxuan Cai, Huimin Zhang, Duancheng Zhao, Jingxing Wu, and Ling Wang. Fp-gnn: a versatile deep learning architecture for enhanced molecular property prediction. *Briefings in bioinformatics*, 23(6):bbac408, 2022.

[6] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, pages 1725–1735. PMLR, 2020.

[7] Na Chen, Ping Li, Jincheng Huang, and Kai Zhang. Denoising structure against adversarial attacks on graph representation learning. *ACM Transactions on Intelligent Systems and Technology*, 16(3):1–23, 2025.

[8] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. Graph trend filtering networks for recommendation. In *SIGIR*, pages 112–121, 2022.

[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *Proceedings of the Web Conference*, pages 417–426, 2019.

[10] Ruiyi Fang, Bingheng Li, Zhao Kang, Qiuhao Zeng, Nima Hosseini Dashtbayaz, Ruizhi Pu, Boyu Wang, and Charles Ling. On the benefits of attribute-driven graph domain adaptation. In *ICLR*, 2025.

[11] Ruiyi Fang, Bingheng Li, Jingyu Zhao, Ruizhi Pu, Qiuhao Zeng, Gezheng Xu, Charles Ling, and Boyu Wang. Homophily enhanced graph domain adaptation. *Forty-Second International Conference on Machine Learning*, 2025.

[12] Ruiyi Fang, Shuo Wang, Ruizhi Pu, Qiuhao Zeng, Hao Zheng, Ziyan Wang, Jiale Cai, Zhimin Mei, Song Tang, Charles Ling, et al. Graph domain adaptation via homophily-agnostic reconstructing structure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 21047–21055, 2026.

[13] Ruiyi Fang, Liangjian Wen, Zhao Kang, and Jianzhuang Liu. Structure-preserving graph representation learning. In *ICDE*, pages 927–932. IEEE, 2022.

[14] Yujie Fang, Xin Li, Qianyu Chen, and Mingzhong Wang. Improving gnn calibration with discriminative ability: insights and strategies. In *AAAI*, volume 38, pages 11953–11960, 2024.

[15] Dominik Fuchsgruber, Tom Wollschläger, and Stephan Günnemann. Energy-based epistemic uncertainty for graph neural networks. In *NeurIPS*, volume 37, pages 34378–34428, 2024.

[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, pages 1321–1330. PMLR, 2017.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1025–1035, 2017.

[18] Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? In *NeurIPS*, volume 35, pages 13775–13786, 2022.

[19] Jincheng Huang, Lun Du, Xu Chen, Qiang Fu, Shi Han, and Dongmei Zhang. Robust mid-pass filtering graph convolutional networks. In *Proceedings of the Web Conference*, pages 328–338, 2023.

[20] Jincheng Huang, Ping Li, Rui Huang, Na Chen, and Acong Zhang. Revisiting the role of heterophily in graph representation learning: An edge classification perspective. *ACM Transactions on Knowledge Discovery from Data*, 18:13:1–13:17, 2023.

[21] Jincheng Huang, Yujie Mo, Ping Hu, Xiaoshuang Shi, Shangbo Yuan, Zeyu Zhang, and Xiaofeng Zhu. Exploring the role of node diversity in directed graph representation learning. In *IJCAI*, pages 2072–2080, 2024.

[22] Jincheng Huang, Yujie Mo, Xiaoshuang Shi, Lei Feng, and Xiaofeng Zhu. Enhancing the influence of labels on unlabeled nodes in graph convolutional networks. In *ICML*, 2025.

[23] Jincheng Huang, Jialie Shen, Xiaoshuang Shi, and Xiaofeng Zhu. On which nodes does GCN fail? enhancing GCN from the node perspective. In *ICML*, 2024.

[24] Kexin Huang, Ying Jin, Emmanuel Candes, and Jure Leskovec. Uncertainty quantification over graph with conformalized graph neural networks. In *NeurIPS*, volume 36, pages 26699–26721, 2023.

[25] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *SIGKDD*, pages 66–74, 2020.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[27] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[28] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *NeurIPS*, volume 32, 2019.

[29] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *ICML*, pages 2805–2814. PMLR, 2018.

[30] Yujie Mo, Zhihe Lu, Runpeng Yu, Xiaofeng Zhu, and Xinchao Wang. Revisiting self-supervised heterogeneous graph learning from spectral clustering perspective. In *NeurIPS*, 2024.

[31] Yujie Mo, Runpeng Yu, Xiaofeng Zhu, and Xinchao Wang. Hg-adapter: Improving pre-trained heterogeneous graph neural networks with dual adapters. In *ICLR*, 2024.

[32] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, volume 29, 2015.

[33] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *ICML*, pages 625–632, 2005.

[34] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

[35] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.

[36] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[37] Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra order-preserving functions for calibration of multi-class neural networks. In *NeurIPS*, volume 33, pages 13456–13467, 2020.

[38] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[39] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[40] Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *NeurIPS*, volume 34, pages 18033–18048, 2021.

[41] Boshi Tang, Zhiyong Wu, Xixin Wu, Qiaochu Huang, Jun Chen, Shun Lei, and Helen Meng. Simcalib: Graph neural network calibration based on similarity between nodes. In *AAAI*, volume 38, pages 15267–15275, 2024.

[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*. OpenReview.net, 2018.

[43] Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. Rethinking calibration of deep neural networks: Do not be afraid of overconfidence. In *NeurIPS*, volume 34, pages 11809–11820, 2021.

[44] Min Wang, Hao Yang, and Qing Cheng. Gcl: Graph calibration loss for trustworthy graph neural network. In *ACM MM*, pages 988–996, 2022.

[45] Min Wang, Hao Yang, Jincai Huang, and Qing Cheng. Moderate message passing improves calibration: A universal way to mitigate confidence bias in graph neural networks. In *AAAI*, volume 38, pages 21681–21689, 2024.

[46] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. In *NeurIPS*, volume 34, pages 23768–23779, 2021.

[47] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

[48] Qitian Wu, Yiting Chen, Chenxiao Yang, and Junchi Yan. Energy-based out-of-distribution detection for graph neural networks. In *ICLR*, 2023.

[49] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5453–5462. PMLR, 2018.

[50] Cheng Yang, Chengdong Yang, Chuan Shi, Yawen Li, Zhiqiang Zhang, and Jun Zhou. Calibrating graph neural networks from a data-centric perspective. In *Proceedings of the ACM on Web Conference*, pages 745–755, 2024.

[51] Xihong Yang, Yue Liu, Sihang Zhou, Siwei Wang, Wenxuan Tu, Qun Zheng, Xinwang Liu, Liming Fang, and En Zhu. Cluster-guided contrastive graph clustering network. In *AAAI*, volume 37, pages 10834–10842, 2023.

[52] Xihong Yang, Cheng Tan, Yue Liu, Ke Liang, Siwei Wang, Sihang Zhou, Jun Xia, Stan Z Li, Xinwang Liu, and En Zhu. Convert: Contrastive graph clustering with reliable augmentation. In *MM*, pages 319–327, 2023.

[53] Xihong Yang, Yiqi Wang, Jin Chen, Wenqi Fan, Xiangyu Zhao, En Zhu, Xinwang Liu, and Defu Lian. Dual test-time training for out-of-distribution recommender system. *IEEE Transactions on Knowledge and Data Engineering*, 37(6):3312–3326, 2025.

[54] Xueying Yang, Jiamian Wang, Xujiang Zhao, Sheng Li, and Zhiqiang Tao. Calibrate automated graph neural network via hyperparameter uncertainty. In *CIKM*, pages 4640–4644, 2022.

[55] Zi Ye, Yogan Jaya Kumar, Goh Ong Sing, Fengyan Song, and Junsong Wang. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741, 2022.

[56] Wenhao You, Bryan Hooi, Yiwei Wang, Euijin Choo, Ming-Hsuan Yang, Junsong Yuan, Zi Huang, and Yujun Cai. Lost in Edits? A λ-Compass for AIGC Provenance. *arXiv preprint arXiv:2502.04364*, 2025.

[57] Acong Zhang, Jincheng Huang, Ping Li, and Kai Zhang. Building shortcuts between distant nodes with biaffine mapping for graph convolutional networks. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–21, 2024.

[58] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. In *NeurIPS*, volume 33, pages 12827–12836, 2020.

[59] Yonghua Zhu, Jincheng Huang, Yang Chen, Robert Amor, and Michael Witbrock. A graph transformer defence against graph perturbation by a flexible-pass filter. *Information Fusion*, 107:102296, 2024.

[60] Dingyi Zhuang, Chonghe Jiang, Yunhan Zheng, Shenhao Wang, and Jinhua Zhao. Gets: Ensemble temperature scaling for calibration in graph neural networks. In *ICLR*, 2025.

A Related Work

This section briefly reviews the topics related to this work, including graph neural networks, Uncertainty Quantification for GNNs, and confidence calibration for GNNs.

A.1 Graph Neural Networks

Graph Neural Networks (GNNs) [51, 27, 42, 19, 52, 20, 23, 31, 30, 13, 10, 22, 21, 7, 57, 59] have emerged as powerful tools for modeling and analyzing graph-structured data. Early approaches, such as the Graph Convolutional Network (GCN) [27], leveraged spectral methods to define convolution operations on graphs, enabling effective semi-supervised learning on citation networks. Subsequent models, including GraphSAGE [17] and GAT [42], introduced sampling-based and attention mechanisms, respectively, to enhance scalability and expressiveness.

However, there are many problems in the basic research and practical application of GNN. For example, over-smoothing and robustness. Subsequent GNN works focus on solving the shortcomings of GNN. To address the challenges of over-smoothing and limited label utilization, recent advancements such as JK-Nets [49] and GCNII [6] proposed architectural modifications to preserve the previous layers' representation information for deep layer architecture and improve the performance of GNNs. For robustness issues, recent studies have also shown that GCNs are vulnerable to adversarial attacks. To solve this issue, recent studies proposed many robust GNNs. For example, Pro-GNN [25] de-noises graphs by constraining graph data to be low-rank, sparse, and feature smoothing. Mid-GCN [19] find that mid-frequency signal in the spectral domain of the graph shows strong robustness, thus designing a mid-pass filtering GNN. Building on these advancements, an emerging line of research focuses on graph calibration, aiming to ensure that the model's confidence aligns with its predictive accuracy, which is crucial for trustworthy deployment in real-world applications.

A.2 Uncertainty Quantification for GNNs

Recent advances have explored uncertainty estimation in graph neural networks (GNNs) from complementary perspectives. GCN-GEBM [15] and GKDE [58] model epistemic and aleatoric uncertainties from an energy-based viewpoint, where the uncertainty is quantified through structure-aware energy potentials or Dirichlet-based evidence distributions, improving robustness under semi-supervised and OOD settings. CF-GNN [24] is the first to extend conformal prediction to graphs, providing theoretical coverage guarantees via calibrated prediction sets. However, these approaches mainly focus on uncertainty quantification or coverage assurance, and their effectiveness on confidence calibration remains underexplored.

A.3 Confidence Calibration for GNNs

Early research on confidence calibration primarily focused on the fields of computer vision and natural language processing [16, 33, 37, 43, 29, 56, 53, 11, 12]. [16] revealed that deep neural networks are often poorly calibrated and investigated the factors influencing calibration. Platt Scaling [36], originally proposed for binary classification models, adjusts raw model outputs into reliable class probability distributions by learning parameters optimized on a validation dataset. To calibrate on multi-class scenarios. Temperature Scaling (TS) [16] proposed to directly modify the model's output probabilities by introducing a temperature coefficient and fine-tuning on a validation set, which is the basis for much subsequent work.

Graph Calibration has emerged in recent years, and CaGCN [46] was the first to discover that GNNs are under-confidence, which is very different from other modern neural networks, which are generally known to be over-confident. Therefore, calibration on GNNs needs more specific approaches. Current graph calibration methods can be classified into two categories, *i.e.,* regularization methods and post-hoc methods. Regularization methods produce calibration regularization terms when training GNNs. For example, GPN [40] calibrates GNNs by performing Bayesian posterior updates for predictions on interdependent nodes. GCL [44] achieves calibration by adding a minimal-entropy regularizer to the KL divergence. HyperU-GCN [54] further incorporates hyperparameter-level uncertainty into Auto-GNNs to improve calibration during training, but it requires additional optimization and model tuning. Post-hoc methods adjust the confidence when model training is completed, and current graph post-hoc methods follow a common paradigm that an additional calibration GNN is trained on the validation set to learn a suitable temperature coefficient mapping, which is then applied via TS for calibration test nodes. For example, CaGCN [46] trains another calibration GNN model on the validation set by using the output probabilities as the input to learn the coefficient of temperature scaling to adjust the confidence, while GATS [18] considers more factors as input, such as neighborhood similarities and homophily. SimCalib [41] relies on node similarity and homophily as input to the calibration

GNN to learn the temperature coefficient. DCGNN [14] designs the self-loop same-class-neighbor ratio as the factor to input to the calibration GNN, improving calibration and discriminative ability. Otherwise, there is a data-centric method, DCGC [50], that can improve the existing post hoc method, which observed that a higher homophily ratio leads to better graph structure calibration. To leverage this, it learns a new homophily-based graph structure to enhance calibration. GETS [60] leverages a mixture-of-experts (MoE) architecture with diverse input transformations, and has demonstrated strong performance on graph calibration tasks. Although the above methods achieve excellent results, they were all designed by introducing extra components for calibration, while neglecting an in-depth exploration of the intrinsic relationship between the model itself and miss-calibration.

B Proofs in Section 3

B.1 Proof of Theorem 3.1

Theorem B.1. *Given the learning rate (*i.e., η*), the final-layer parameters (*i.e., $\mathbf{W}^{(K)}$*). For an arbitrary node v in model training stage, its output probabilities on class i (*i.e., $s_{v,i}$*) is updated by:*

$$s'_{v,i} = \frac{e^{b_{v,i}/\tau}}{e^{b_{v,i}/\tau} + \sum_{j\neq i}^{c} e^{b_{v,j}/\tau} \cdot \psi_{i,j}}$$
$$s.t., \begin{cases} \psi_{i,j} = e^{\eta(s_{v,i} - y_{v,i} - s_{v,j} + y_{v,j})(\mathbf{h}_v^{(K)})^\top \mathbf{h}_v'^{(K)}} \\ \tau = \frac{1}{1-\eta\lambda^{(K)}} \\ b_{v,i} = (\mathbf{W}_{:,i}^{(K)})^\top \mathbf{h}_v'^{(K)} \end{cases}, \tag{11}$$

where the value and representation $z'_{v,i}$ and $\mathbf{h}_v'^{(K)}$ are the updated results of $z_{v,i}$ and $\mathbf{h}_v^{(K)}$ after the next epoch.

Proof. Given the CE and weight decay:

$$\mathcal{L}_v = -\sum_{i=1}^{c} y_{v,i} \log s_{v,i} + \sum_{k}^{K} \frac{\lambda^{(k)}}{2} ||\mathbf{W}^{(k)}||_F^2, \tag{12}$$

Then we can calculate the derivative of the final-layer parameters (*i.e.,* $\mathbf{W}^{(K)} \in \mathbb{R}^{d\times c}$):

$$\frac{\partial \mathcal{L}_v}{\partial \mathbf{W}_{:,i}^{(K)}} = \sum_{j=1}^{c} (\frac{\partial \mathcal{L}_v}{\partial s_{v,j}} \frac{\partial s_{v,j}}{\partial z_{v,i}}) \frac{\partial z_{v,i}}{\partial \mathbf{W}_{:,i}^{(K)}} + \frac{\partial \sum_{k=1}^{K} \frac{\lambda^{(k)}}{2} \left\|\mathbf{W}^{(k)}\right\|_F^2}{\partial \mathbf{W}_{:,i}^{(K)}} = (s_{v,i} - y_{v,i})\mathbf{h}_v^{(K)} + \lambda^{(K)} \mathbf{W}_{:,i}^{(K)}. \tag{13}$$

The details of derivation are listed in Appendix B.1.1. Then the final-layer parameter $\mathbf{W}^{(K)}$ is updated as:

$$\mathbf{W}_{:,i}'^{(K)} = (1 - \eta\lambda^{(K)})\mathbf{W}_{:,i}^{(K)} - \eta(s_{v,i} - y_{v,i})\mathbf{h}_v^{(K)}, \tag{14}$$

where $\mathbf{W}_{:,i}'^{(K)}$ is the next epoch of $\mathbf{W}_{:,i}^{(K)}$ and then the output logits $z_{v,i}$ rely on $\mathbf{W}^{(K)}$ is updated as:

$$\begin{aligned} z'_{v,i} &= (\mathbf{h}_v'^{(K)})^\top \mathbf{W}_{:,i}'^{(K)} \\ &= (1 - \eta\lambda^{(K)})(\mathbf{h}_v'^{(K)})^\top \mathbf{W}_{:,i}^{(K)} - \eta(s_{v,i} - y_{v,i})(\mathbf{h}_v'^{(K)})^\top \mathbf{h}_v^{(K)}, \end{aligned} \tag{15}$$

where $\mathbf{h}_v'^{(l)}$ and $z'_{v,i}$ is the next epoch of $\mathbf{h}_v^{(l)}$ and $z_{v,i}$, respectively. Furthermore, the output probabilities can be obtained:

$$\begin{aligned} s_{v,i} &= \frac{e^{z'_{v,i}}}{e^{z'_{v,i}} + \sum_{k\neq i}^{c} e^{z'_{v,k}}} \\ &= \frac{e^{(1-\eta\lambda^{(K)})(\mathbf{h}_v'^{(l)})^\top \mathbf{W}_{:,i}^{(K)} - \eta(s_{v,i} - y_{v,i})(\mathbf{h}_v'^{(l)})^\top \mathbf{h}_v^{(l)}}}{e^{(1-\eta\lambda^{(K)})(\mathbf{h}_v'^{(l)})^\top \mathbf{W}_{:,i}'^{(K)} - \eta(s_{v,i} - y_{v,i})(\mathbf{h}_v'^{(l)})^\top \mathbf{h}_v^{(l)}} + \sum_{j\neq i}^{c} e^{(1-\eta\lambda^{(K)})(\mathbf{h}_v'^{(l)})^\top \mathbf{W}_{:,j}^{(K)} - \eta(s_{v,j} - y_{v,j})(\mathbf{h}_v'^{(l)})^\top \mathbf{h}_v^{(l)}}}. \end{aligned} \tag{16}$$

We let $\tau = \frac{1}{1-\eta\lambda^{(K)}}$ and $b_i = (\mathbf{h}_v'^{(l)})^\top \mathbf{W}_{:,i}^{(K)}$, then we have:

$$\begin{aligned} s_{v,i} &= \frac{e^{b_i/\tau} / e^{\eta(s_{v,i}-y_{v,i})(\mathbf{h}_v'^{(l)})^T \mathbf{h}_v^{(l)}}}{e^{b_i/\tau} / e^{\eta(s_{v,i}-y_{v,i})(\mathbf{h}_v'^{(l)})^T \mathbf{h}_v^{(l)}} + \sum_{j\neq i}^{c} e^{b_j/\tau} / e^{\eta(s_{v,j}-y_{v,i})(\mathbf{h}_v'^{(l)})^T \mathbf{h}_v^{(l)}}}. \\ &= \frac{e^{b_i/\tau}}{e^{b_i/\tau} + \sum_{j\neq i}^{c} e^{b_j/\tau}.e^{\eta(s_{v,i}-y_{v,i}-s_{v,j}+y_{v,j})(\mathbf{h}_v^{(K)})^\top \mathbf{h}_v'^{(K)}}}. \end{aligned} \tag{17}$$

We let $\psi_{i,j} = e^{\eta(s_{v,i}-y_{v,i}-s_{v,j}+y_{v,j})(\mathbf{h}_v^{(K)})^\top \mathbf{h}_v'^{(K)}}$, thus, we can get the finally form:

$$s'_{v,i} = \frac{e^{b_{v,i}/\tau}}{e^{b_{v,i}/\tau} + \sum_{j\neq i}^{c} e^{b_{v,j}/\tau}.\psi_{i,j}} \tag{18}$$

□

B.1.1 derivation of Eq.(13)

Proof.

$$\frac{\partial \mathcal{L}_v}{\partial \mathbf{W}_{:,i}^{(K)}} = \sum_{j=1}^{c} (\frac{\partial \mathcal{L}_v}{\partial s_{v,j}} \frac{\partial s_{v,j}}{\partial z_{v,i}}) \frac{\partial z_{v,i}}{\partial \mathbf{W}_{:,i}^{(K)}} + \frac{\partial \sum_{k=1}^{K} \frac{\lambda^{(k)}}{2} \left\| \mathbf{W}^{(k)} \right\|_F^2}{\partial \mathbf{W}_{:,i}^{(K)}} \tag{19}$$

We can calculate the derivatives of each part separately.

$$\frac{\partial \mathcal{L}}{\partial s_{v,j}} = \frac{\partial \sum_{i=1}^{c} y_{v,i} \log s_{v,i} + \lambda^{(K)} \sum_{k}^{K} ||\mathbf{W}^{(k)}||_F^2}{\partial s_{v,j}} = -\frac{y_{v,j}}{s_{v,j}}. \tag{20}$$

For $\frac{\partial s_{v,j}}{\partial z_{v,i}}$, the forward equation is softmax function:

$$s_{v,j} = \frac{e^{z_{v,j}}}{e^{z_{v,j}} + \sum_{k\neq j}^{c} e^{z_{v,k}}}. \tag{21}$$

There are two cases of derivation here. In the first case, when $j = i$, the independent variable appears in both the numerator and the denominator, we have

$$\begin{aligned} \frac{\partial \frac{e^{z_{v,i}}}{\sum_k^c e^{z_{v,k}}}}{\partial z_{v,i}} &= -\frac{e^{z_{v,i}}.e^{z_{v,i}}}{(\sum_{k=1}^{c} e^{z_{v,k}})^2} + \frac{e^{z_{v,i}}}{\sum_{k=1}^{c} e^{z_{v,k}}} \\ &= \frac{e^{z_{v,i}}}{\sum_{k=1}^{c} e^{z_{v,k}}} (1 - \frac{e^{z_{v,i}}}{\sum_{k=1}^{c} e^{z_{v,k}}}) \\ &= s_{v,i}(1 - s_{v,i}). \end{aligned} \tag{22}$$

When $j \neq i$, the independent variable appears only in the denominator, it is easy to obtain:

$$\frac{\partial \frac{e^{z_{v,j}}}{\sum_k^c e^{z_{v,k}}}}{\partial z_{v,i}} = -\frac{e^{z_{v,j}}}{\sum_k^c e^{z_{v,k}}} . \frac{e^{z_{v,i}}}{\sum_k^c e^{z_{v,k}}} = -s_{v,k} s_{v,i}. \tag{23}$$

Finally, we have

$$\begin{aligned} \frac{\partial \mathcal{L}_v}{\partial \mathbf{W}_{:,i}^{(K)}} &= (\sum_{k\neq i}^{c} \frac{y_{v,k}}{s_{v,k}} . s_{v,k} . s_{v,i} - \frac{y_{v,i}}{s_{v,i}} . s_{v,i} . (1 - s_{v,i})) \frac{\partial s_{v,i}}{\mathbf{W}_{:,i}^{(K)}} + \frac{\partial \lambda^{(K)} \sum_{k}^{K} ||\mathbf{W}^{(k)}||_F^2}{\mathbf{W}_{:,i}^{(K)}} \\ &= (s_{v,i} - y_{v,i}) \mathbf{h}_v^{(K)} + \lambda^{(K)} \mathbf{W}_{:,i}^{(K)} \end{aligned} . \tag{24}$$

□

B.2 Proof for Theorem 3.2

Theorem B.2. *(Optimality condition for $\mathbf{W}^{(K)}$) Given the objective function Eq. (4) with $\mathbf{s}_v = \mathrm{softmax}(\mathbf{W}^{(K)}\mathbf{h}_v^{(K)})$, the optimizer $\mathbf{W}^{(K)*}$ satisfies the following stationary condition. For each class $i \in \{1, \dots, c\}$,*

$$(\mathbf{W}_{:,i}^{(K)})^* = \frac{1}{\lambda^{(K)}}\Big(\sum\nolimits_{u\in\mathcal{T}:y_{u,i}=1}(1-s_{u,i}^*)\mathbf{h}_u^{(K)} - \sum\nolimits_{v\in\mathcal{T}:y_{v,i}\neq 1} s_{v,i}^*\mathbf{h}_v^{(K)}\Big), \tag{25}$$

where s^* denotes the softmax probability evaluated at the optimal solution $\mathbf{W}^{(K)*}$.

Proof. Let $\mathcal{T}$ denote the set of training nodes and consider the overall objective $\mathcal{L} = \sum_{v\in\mathcal{T}}\mathcal{L}_v$ (Eq. (4)). For a fixed class index i, using the standard gradient of softmax cross-entropy (see Appendix B.1), we have

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{:,i}^{(K)}} = \sum\nolimits_{v\in\mathcal{T}}(s_{v,i}(\mathbf{W}^{(K)}) - y_{v,i})\mathbf{h}_v^{(K)} + \lambda^{(K)}\mathbf{W}_{:,i}^{(K)}. \tag{26}$$

At the optimum $\mathbf{W}^{(K)*}$, the first-order necessary (and here sufficient) condition gives $\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{:,i}^{(K)}}\big|_{\mathbf{W}^{(K)}=\mathbf{W}^{(K)*}} = \mathbf{0}$. Substituting into Eq. (26) yields

$$\lambda^{(K)}(\mathbf{W}_{:,i}^{(K)})^* = \sum\nolimits_{v\in\mathcal{T}}(y_{v,i} - s_{v,i}^*)\mathbf{h}_v^{(K)}. \tag{27}$$

Finally, splitting the summation according to whether $y_{v,i} = 1$ or $y_{v,i} \neq 1$ leads to

$$\lambda^{(K)}(\mathbf{W}_{:,i}^{(K)})^* = \sum\nolimits_{u\in\mathcal{T}:y_{u,i}=1}(1-s_{u,i}^*)\mathbf{h}_u^{(K)} - \sum\nolimits_{v\in\mathcal{T}:y_{v,i}\neq 1} s_{v,i}^*\mathbf{h}_v^{(K)}, \tag{28}$$

which proves Eq. (25).

Remark. Since $s_{v,i}^*$ depends on $(\mathbf{W}^{(K)})^*$ through the softmax mapping, Eq. (25) is an implicit (fixed-point) optimality condition rather than a closed-form expression. □

B.3 Proof for Proposition 3.3

Proposition B.3. *Consider the final-layer optimization problem with fixed final-layer representations: $\mathbf{W}_\lambda^* = \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \frac{\lambda}{2}\|\mathbf{W}\|_F^2$, where $\mathcal{L}(\mathbf{W})$ is the cross-entropy loss and λ is the weight decay coefficient of the final-layer parameters. Let $\mathbf{w}_i^*(\lambda)$ denote the optimized weight vector of class i. For two coefficients $\lambda_1 > \lambda_2 > 0$, the average squared distance between final-layer class prototypes satisfies:*

$$\frac{1}{c(c-1)}\sum_{i\neq j}\|\mathbf{w}_i^*(\lambda_1) - \mathbf{w}_j^*(\lambda_1)\|_2^2 \leq \frac{1}{c(c-1)}\sum_{i\neq j}\|\mathbf{w}_i^*(\lambda_2) - \mathbf{w}_j^*(\lambda_2)\|_2^2. \tag{29}$$

Therefore, reducing the final-layer weight decay enlarges the overall distance among final-layer class prototypes.

Proof. Let $\mathbf{W}_1 = \mathbf{W}_{\lambda_1}^*$ and $\mathbf{W}_2 = \mathbf{W}_{\lambda_2}^*$, where $\lambda_1 > \lambda_2 > 0$. By the optimality of $\mathbf{W}_1$ under λ_1, we have

$$\mathcal{L}(\mathbf{W}_1) + \frac{\lambda_1}{2}\|\mathbf{W}_1\|_F^2 \leq \mathcal{L}(\mathbf{W}_2) + \frac{\lambda_1}{2}\|\mathbf{W}_2\|_F^2. \tag{30}$$

Similarly, by the optimality of $\mathbf{W}_2$ under λ_2, we have

$$\mathcal{L}(\mathbf{W}_2) + \frac{\lambda_2}{2}\|\mathbf{W}_2\|_F^2 \leq \mathcal{L}(\mathbf{W}_1) + \frac{\lambda_2}{2}\|\mathbf{W}_1\|_F^2. \tag{31}$$

Adding the above two inequalities gives

$$\frac{\lambda_1 - \lambda_2}{2}\|\mathbf{W}_1\|_F^2 \leq \frac{\lambda_1 - \lambda_2}{2}\|\mathbf{W}_2\|_F^2. \tag{32}$$

Since $\lambda_1 > \lambda_2$, it follows that

$$\|\mathbf{W}_{\lambda_1}^*\|_F \leq \|\mathbf{W}_{\lambda_2}^*\|_F. \tag{33}$$

Next, we connect the norm of the final-layer weights to the distances between class prototypes. For the final-layer softmax classifier, adding the same vector to all class weight vectors does not change the predicted probabilities, since it adds the same scalar to all logits. Under ℓ_2 regularization, the minimum-norm representative is the centered one, i.e.,

$$\sum_{i=1}^{c} \mathbf{w}_i^* = \mathbf{0}. \tag{34}$$

Using the standard identity for pairwise distances, we have

$$\sum_{i<j} \|\mathbf{w}_i^* - \mathbf{w}_j^*\|_2^2 = c \sum_{i=1}^{c} \|\mathbf{w}_i^*\|_2^2 - \left\| \sum_{i=1}^{c} \mathbf{w}_i^* \right\|_2^2. \tag{35}$$

Since $\sum_{i=1}^{c} \mathbf{w}_i^* = \mathbf{0}$, this reduces to

$$\sum_{i<j} \|\mathbf{w}_i^* - \mathbf{w}_j^*\|_2^2 = c\|\mathbf{W}^*\|_F^2. \tag{36}$$

Equivalently,

$$\frac{1}{c(c-1)} \sum_{i \neq j} \|\mathbf{w}_i^* - \mathbf{w}_j^*\|_2^2 = \frac{2}{c-1} \|\mathbf{W}^*\|_F^2. \tag{37}$$

Therefore, since $\|\mathbf{W}_{\lambda_1}^*\|_F \leq \|\mathbf{W}_{\lambda_2}^*\|_F$, we obtain

$$\frac{1}{c(c-1)} \sum_{i \neq j} \|\mathbf{w}_i^*(\lambda_1) - \mathbf{w}_j^*(\lambda_1)\|_2^2 \leq \frac{1}{c(c-1)} \sum_{i \neq j} \|\mathbf{w}_i^*(\lambda_2) - \mathbf{w}_j^*(\lambda_2)\|_2^2. \tag{38}$$

This completes the proof. □

B.4 Proof of Proposition 3.4

Proposition B.4 (Effect of Message Passing)**.** *Consider a K-layer linearized GNN $\mathbf{H}^{(K)} = \mathbf{P}^K \mathbf{X} \mathbf{\Theta}$ with a nonnegative row-normalized propagation matrix $\mathbf{P}$, where $P_{vu} > 0$ only if $u \in \mathcal{N}_v \cup \{v\}$. Let $\mathbf{w}_i = \mathbf{W}_{:,i}^{(K)}$ be the prototype of class i, $\mathcal{T}_i$ the training nodes of class i, and $\rho_{v,i}^{(K)} = \sum_{u \in \mathcal{T}_i} [\mathbf{P}^K]_{vu}$ the same-class training influence on node v. Then,*

$$\|\mathbf{h}_v^{(K)} - \mathbf{w}_i\|_2 \leq R_i - \rho_{v,i}^{(K)}(R_i - \delta_i),$$

where $\delta_i < R_i$ denote the corresponding prototype-deviation bounds. Hence, weaker same-class training influence leads to a larger upper bound on the prototype deviation. Moreover, $d_{\mathcal{G}}(v, \mathcal{T}_i) > K$ implies $\rho_{v,i}^{(K)} = 0$.

Proof. Let $\mathbf{x}_u^\top$ and $\mathbf{h}_v^{(K)\top}$ denote the u-th and v-th rows of $\mathbf{X}$ and $\mathbf{H}^{(K)}$, respectively. From $\mathbf{H}^{(K)} = \mathbf{P}^K \mathbf{X} \mathbf{\Theta}$, we have

$$\mathbf{h}_v^{(K)} = \sum_{u \in \mathcal{V}} [\mathbf{P}^K]_{vu} \mathbf{\Theta}^\top \mathbf{x}_u. \tag{39}$$

Class smoothness. We assume that the class-i training nodes are closer to their corresponding prototype $\mathbf{w}_i$ than the remaining nodes, i.e.,

$$\|\mathbf{\Theta}^\top \mathbf{x}_u - \mathbf{w}_i\|_2 \leq \begin{cases} \delta_i, & u \in \mathcal{T}_i, \\ R_i, & u \notin \mathcal{T}_i, \end{cases} \qquad \delta_i < R_i. \tag{40}$$

This is natural since $\mathbf{w}_i$ is determined by the labeled training representations and corresponds to class i, as characterized in Theorem 3.2.

Since $\mathbf{P}$ is nonnegative and row-normalized, $\mathbf{P}^K$ is also nonnegative and row-normalized. Hence, $\sum_{u\in\mathcal{V}}[\mathbf{P}^K]_{vu} = 1$. By the triangle inequality and Eq. (40),

$$\begin{aligned}\|\mathbf{h}_v^{(K)} - \mathbf{w}_i\|_2 &= \left\|\sum_{u\in\mathcal{V}}[\mathbf{P}^K]_{vu}\left(\mathbf{\Theta}^\top \mathbf{x}_u - \mathbf{w}_i\right)\right\|_2 \\ &\le \delta_i \sum_{u\in\mathcal{T}_i}[\mathbf{P}^K]_{vu} + R_i \sum_{u\notin\mathcal{T}_i}[\mathbf{P}^K]_{vu} \\ &= \rho_{v,i}^{(K)}\delta_i + \left(1-\rho_{v,i}^{(K)}\right)R_i \\ &= R_i - \rho_{v,i}^{(K)}(R_i - \delta_i). \end{aligned} \tag{41}$$

Since $R_i > \delta_i$, the upper bound in Eq. (41) decreases monotonically with $\rho_{v,i}^{(K)}$.

We next consider the locality of message passing. Expanding $\mathbf{P}^K$ gives

$$[\mathbf{P}^K]_{vu} = \sum_{w_1,\ldots,w_{K-1}\in\mathcal{V}} P_{vw_1}P_{w_1w_2}\cdots P_{w_{K-1}u}. \tag{42}$$

Since $P_{ab} > 0$ only if $b \in \mathcal{N}_a \cup \{a\}$, each nonzero term corresponds to a walk from v to u within K message-passing steps. Thus,

$$d_{\mathcal{G}}(v,u) > K \quad\Longrightarrow\quad [\mathbf{P}^K]_{vu} = 0. \tag{43}$$

Consequently, if $d_{\mathcal{G}}(v,\mathcal{T}_i) > K$, then $[\mathbf{P}^K]_{vu} = 0$ for every $u \in \mathcal{T}_i$, and therefore

$$\rho_{v,i}^{(K)} = \sum_{u\in\mathcal{T}_i}[\mathbf{P}^K]_{vu} = 0. \tag{44}$$

This completes the proof. □

C Additional Methodological Details

C.1 Schematic of Node-Level Calibration

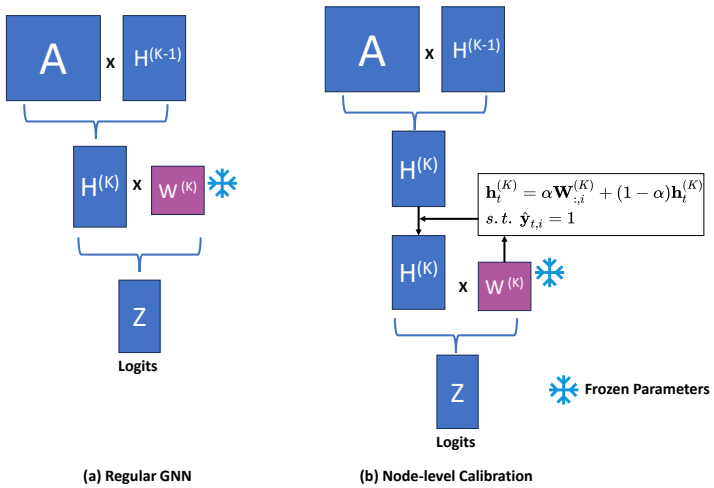


Figure 4: Illustration of the proposed node-level calibration in the final layer during test time.

The Schematic of Node-Level Calibration is shown in Figure 4. The proposed node-level calibration introduces only a lightweight adjustment to the final-layer representation $\mathbf{H}^{(K)}$, guided by the final-layer parameters $\mathbf{W}^{(K)}$ and predicted labels.

D Experimental Settings

This section provides detailed experimental settings in Section 4, including the description of all datasets in Section D.1, summarization of all comparison methods in Section D.2, model architectures and settings in Section D.3, and the evaluation protocol in Section D.4.

D.1 Datasets

We follow the previous works [46] choose the commonly used Cora [39], Citeseer [39], Pubmed [39], and CoraFull [3] for evaluation, where re composed of papers as nodes and their relationships such as citation relationships, and common authoring. Node feature is a one-hot vector that indicates whether a word is present in that paper. Words with a frequency of less than 10 are removed. We choose 500 nodes for validation and 1000 for testing and select three label rates for the training set (*i.e.,* 20, 40, and 60 labeled nodes per class). The details of these datasets are summarized in Table 5. All experiments setting are performed following the official code [46].

Table 5: The statistics of the datasets

Datasets	Nodes	Edges	Train	Valid	Test Nodes	Features	Classes
Cora	2,708	5,429	140/280/420	500	1000	1,433	7
Citeseer	3,327	4,732	120/124/360	500	1,000	3,703	6
Pubmed	19,717	44,338	60/120/360	500	1,000	500	3
CoraFull	19,793	126,842	1400/2800/4200	500	1000	8,710	70

D.2 Comparison Methods

This study focuses on the classical semi-supervised node classification task. To evaluate model calibration, we follow the most authoritative work [46] that considers two widely-used GNN architectures: GCN [27] and GAT [42].

GCN is the most traditional undirected graph neural network, which updates node representations by aggregating neighboring formations. GAT is a graph neural network model that extends the self-attention mechanism to the graph, and updates nodes' representation by attention score to aggregate neighbors.

We compare several post-hoc calibration methods commonly applied to neural networks, including Temperature Scaling (TS) [16] and Matrix Scaling (MS) [28]. Furthermore, the state-of-the-art graph calibration models are also compared:

- **CaGCN** [46], the first calibration method designed specifically for GNNs, which employs a calibration model to scale the confidence of every node after training the mode.
- **GATS** [18] is the same as CaGCN which both learn instance-wise temperatures based on a heuristic formula.
- **AU-LS** [45] is a regularization method for calibrating GNNs, utilizing a reverse label smoothing objective function to enhance the confidence of GNN predictions. We employ the 2-layer GNN variant, as the accuracy of the 3-layer GNN is significantly lower, making further discussion unnecessary. Additionally, the accuracy of the 2-layer AULS variants is also reported in Table E.2.
- **DCGC** [50] is a data-centric calibration method, that learns a homophily graph for better calibration. In the experimental results, we report the results of the best DCGC variant GATS-DCGC.

D.3 Model Architectures and Settings

In our experiments, we follow [46], adopting a two-layer configuration for GCN and GAT, with hidden layer dimensions selected from {8, 16, 64}. All parameters are optimized using the Adam optimizer [26] with the learning rate selected from {0.01, 0.015, 0.02}, dropout selected from {0.5, 0.6}, and the weight decay for all layers is set to 0.0005. In our proposed model, the weight decay of

Table 6: Settings for the proposed SCAR based on GCN.

Datasets	**L/C**	Lr	Weight decay	Hidden units	Dropout	Weight decay of $\mathbf{W}^{(K)}$	α	β
Cora	20	0.015	5e-4	64	0.6	1e-4	2e-4	2e-3
	40	0.015	5e-4	64	0.6	1e-4	1e-4	1e-3
	60	0.015	5e-4	64	0.6	1e-4	1e-4	1e-3
Citeseer	20	0.01	5e-4	64	0.5	4.5e-4	5e-5	5e-3
	40	0.01	5e-4	64	0.5	4.5e-4	5e-5	5e-3
	60	0.01	5e-4	64	0.5	4e-4	5e-4	2e-3
Pubmed	20	0.02	5e-4	16	0.5	4e-4	5e-8	5e-6
	40	0.02	5e-4	16	0.5	4e-4	5e-7	5e-5
	60	0.02	5e-4	16	0.5	4e-4	5e-7	5e-5
CoraFull	20	0.01	5e-4	64	0.5	2e-6	3e-5	5e-4
	40	0.01	5e-4	64	0.5	1e-4	3e-4	5e-3
	60	0.01	5e-4	64	0.5	2e-6	3e-5	5e-4

the final-layer parameter $\mathbf{W}^{(K)}$ is set to smaller than 0.0005 (used for the other layers' parameters). the hyper-parameters α, β are selected from (0.0005, 0.000005] and $\alpha < \beta$. Table 6 and Table 7 describe the detailed settings and architecture for our experimental setups with SCAR based on GCN and GAT backbones, respectively.

Table 7: Settings for the proposed SCAR based on GAT.

Datasets	**L/C**	Lr	Weight decay	Hidden units	Dropout	Weight decay of $\mathbf{W}^{(K)}$	α	β
Cora	20	0.01	5e-4	8	0.5	2e-5	5e-6	4e-4
	40	0.01	5e-4	8	0.5	2e-5	5e-6	4e-4
	60	0.01	5e-4	8	0.5	2e-5	5e-6	4e-4
Citeseer	20	0.01	5e-4	8	0.6	3e-4	5e-4	5e-3
	40	0.01	5e-4	8	0.6	3e-4	5e-4	5e-3
	60	0.01	5e-4	8	0.6	3e-4	5e-4	5e-3
Pubmed	20	0.01	5e-4	8	0.6	3e-4	5e-6	5e-5
	40	0.01	5e-4	6	0.5	3e-4	5e-7	5e-5
	60	0.01	5e-4	8	0.5	3e-4	5e-7	5e-6
CoraFull	20	0.01	5e-4	8	0.6	2e-6	3e-5	5e-4
	40	0.01	5e-4	8	0.6	2e-6	3e-5	5e-4
	60	0.01	5e-4	8	0.6	2e-6	3e-5	5e-4

D.4 Evaluation Protocol

We follow the evaluation in previous works [46], adopting Expected Calibration Error (ECE) with 20 bins as the metric for calibration. Besides, since fine-tuning the hyperparameter of the weight decay in the final layer, the predicted label may have slight changes, we also adopt classification accuracy as an evaluation metric. We report the average and standard deviation of 10 runs for each pair split of a dataset.

D.5 Computing Resource Details

All experiments were implemented in PyTorch and conducted on a server with 8 NVIDIA GeForce 4090 (24 GB memory each). Almost every experiment can be done on an individual 4090, and the training time of all comparison methods as well as our method, is less than 1 hour.

E Additional Experiments

E.1 Analysis of Hyper-parameters

In the proposed method, SCAR, we theoretically prove that the weight decay of the final layer leads to exacerbating the under-confidence of GNNs, thus we propose to reduce the final-layer weight

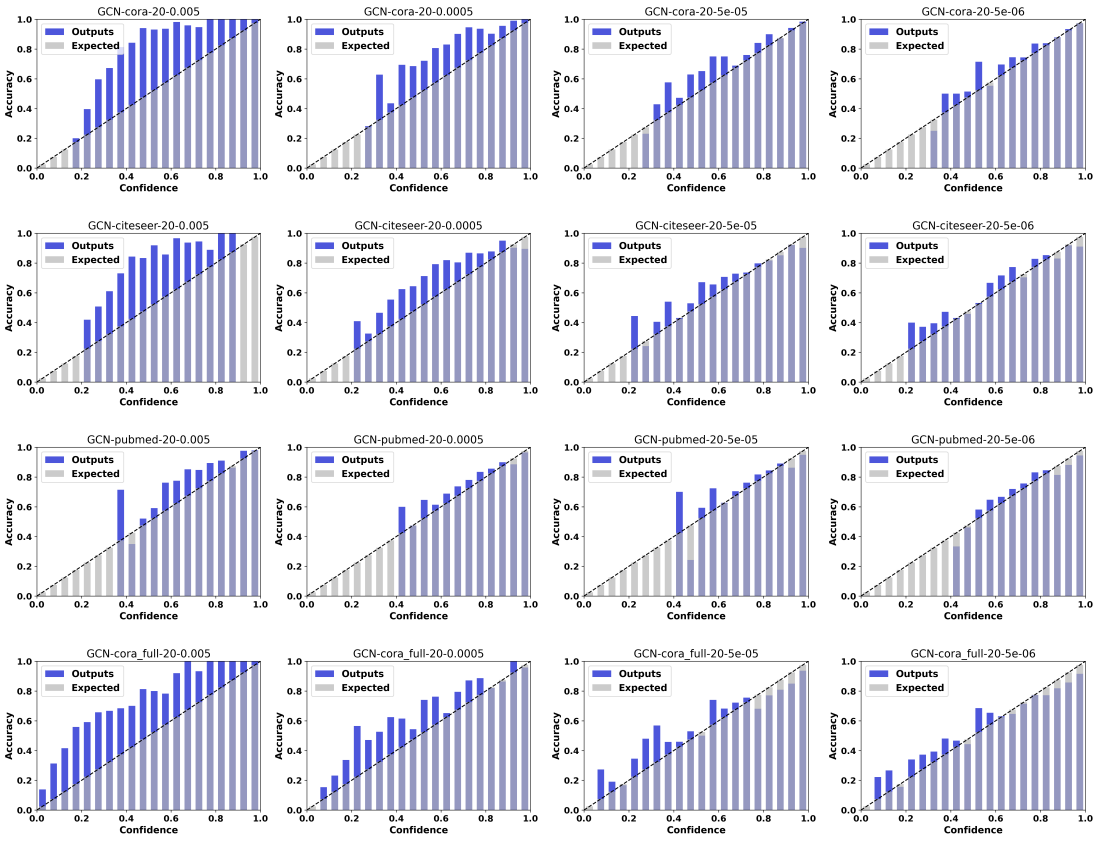
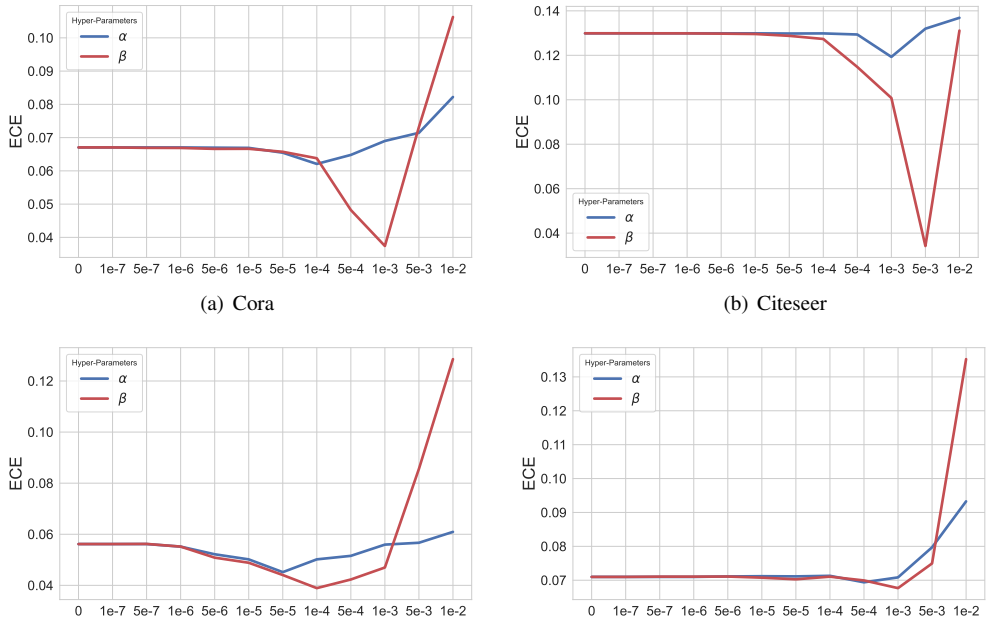


Figure 5: Reliability diagrams under different values of $\lambda^{(K)}$ (0.005, 0.0005, 0.00005, 0.000005) on the Cora, Citeseer, Pubmed, and CoraFull datasets. The base model is GCN and L/C=20.

decay to avoid the under-confidence phenomenon. Moreover, we employ the non-negative parameters (*i.e.,* α and β) to control the similarity between cluster centroid and test nodes, where α controls the test nodes adjacent to the training nodes, and β controls the remaining test nodes.

To investigate the impact of weight decay of the final layer, we conduct node classification on all datasets with L/C=20 by varying the value of the final-layer weight decay in the range of [0.005, 0.0005, 0.00005, 0.000005] (from left to right) and report the results in Figure 5. As shown in Figure 5, it is clearly observed that as the weight decay of the final layer decreases, the model transitions from being under-confident to confident, and its calibration performance improves progressively. This further validates the results of our Theorem 3.1 and highlights the importance of adjusting the weight decay of the final layer for improving confidence. For the selection of the final-layer weight decay, based on this observation, we recommend a binary search strategy, allowing us to efficiently determine the optimal weight decay value.

To investigate the impact of α and β with different settings, we conduct the node classification on all datasets with L/C=20 by fixing a hyper-parameter and varying the value of another hyper-parameter (*i.e.,* both α and β) in the range of [0, 0.001]. Note that we have chosen a suitable weight decay parameter for $\mathbf{W}^{(K)}$. The results are reported in Figure 6. From Figure 6, we have the following observations. First, adjusting β (*i.e.,* the red line) always obtains a lower ECE than α (*i.e.,* the blue line). This indicates that improving the similarity between test nodes disconnected from training nodes and their predicted class prototype is more effective than for connected test nodes. Second, the optimal value of β is always higher than α. This indicates that the test nodes disconnected from training nodes need a larger weight to move closer to the class prototype. The above observations are consistent with our analysis that test nodes farther from the training nodes are more under-confident. For the selection of α and β, Figure 6 already uggests that β should be greater than α. Leveraging this insight, we employed a partial grid search (i.e., searching only within the triangular region where



(a) Cora (b) Citeseer

(c) Pubmed (d) CoraFull

Figure 6: Visualization of the impact of defferent α and β on ECE and red line represnet β, the blue line represent α. The base model is GCN and L/C=20.

$\beta > \alpha$), effectively reducing the search space by half while maintaining thorough exploration of meaningful parameter settings.

E.2 Accuracy Analysis

The proposed SCAR framework comprises two components: class-centroid-level calibration and node-level calibration. Notably, since node-level calibration is a post-hoc method, which does not alter the model's predictions, while only class-centroid-level calibration (reducing the weight decay of the final layer) influences the predictions. Although adjusting the weight decay of the final layer is typically considered part of standard hyperparameter tuning, we conducted experiments to evaluate its impact on accuracy compared with the uncalibrated model and regularization method (*i.e.,* AU-LS). The results are shown in Table 8.

From Table 8, we can observe that across all datasets and L/C configurations, SCAR consistently outperforms the uncalibrated models and regularization model (*i.e.,* AU-LS) for both GCN and GAT backbones. This demonstrates that reducing the weight decay of the final layer not only achieves a good calibration of the GNNs' confidence but also improves their prediction accuracy. It is worth noting that, especially in the CoraFull dataset, the proposed SCAR on average improves by 2.46 % and 5.7 %, compared to GCN and GAT.

E.3 Validation on Heterophilic Dataset

To comprehensively evaluate the effectiveness and generalization of the proposed method, we conduct experiments on a wide range of graph datasets. In addition to commonly used homophilic benchmarks, we further include heterophilic graphs (*i.e.,* Chameleon [34], Squirrel [34], and Arxiv-year [35], wherein Arxiv-year is also a large-scale graphs) to verify the adaptability of our approach under diverse structural and label distribution settings. The experiment results are shown in Table 9.

From Table 9, we can observe that across all heterophilic datasets, the proposed SCAR consistently improves the ECE by a large margin, especially on the large-scale Arxiv-year dataset. Specifically, on Arxiv-year, SCAR reduces the ECE by 54.8% and 51.8% when applied to GCN and GAT, respectively. This demonstrates the generality of our method across diverse types of graph datasets.

Table 8: Accuracy of node classification (%), considering various numbers of labels per class (L/C). *Uncal.* represents the uncalibrated model. The best results are highlighted in black.

Datasets	L/C	GCN			GAT		
		Uncal.	**AU-LS**	**SCAR**	**Uncal.**	**AU-LS**	**SCAR**
Cora	20	81.53±0.32	79.58±0.14	**82.01±0.21**	82.56±0.31	80.53±1.05	**83.45±0.29**
	40	**83.04±0.24**	82.90±0.69	82.70±0.16	**83.56±0.25**	82.28±0.99	83.35±0.42
	60	84.35±0.35	84.03±0.80	**84.49±0.37**	85.06±0.31	84.55±0.61	**85.07±0.53**
Citeseer	20	**71.71±0.32**	65.65±2.91	71.61±0.45	71.47±0.88	68.57±1.01	**71.51±0.60**
	40	72.18±0.32	70.71±1.62	**72.43±0.41**	72.31±0.56	71.55±0.47	**72.37±0.31**
	60	72.98±0.25	72.91±0.49	**73.01±0.50**	72.91±0.69	73.15±0.38	**73.23±0.51**
Pubmed	20	79.53±0.13	76.74±2.61	**79.60±0.40**	**78.88±0.31**	77.92±0.56	78.62±0.36
	40	80.57±0.34	78.59±0.68	**80.84±0.23**	**80.64±0.43**	79.26±0.30	80.21±0.57
	60	83.11±0.24	80.51±0.51	**83.24±0.54**	**82.27±0.30**	80.79±0.36	82.07±0.37
CoraFull	20	62.03±0.27	61.62±0.62	**64.17±0.52**	58.87±0.35	54.71±0.43	**62.91±0.32**
	40	65.02±0.32	62.31±0.67	**67.21±0.46**	60.08 ±0.30	58.39±0.48	**66.17±0.48**
	60	66.71±0.33	64.12±0.83	**69.77±0.31**	61.39±0.27	59.37±0.52	**68.38±0.37**

Table 9: ECE (%) with 20 bins on different models for heterophilic datasets, where the Arxiv-year dataset is also included as a representative large-scale graph. The best results are highlighted in black.

Datasets	Chameleon	Squirrel	Arxiv-year
GCN	13.82±0.75	11.67±0.18	13.34±0.72
SCAR-GCN	**7.30±1.48**	**7.25±0.35**	**6.03±0.38**
GAT	14.11±0.23	11.13±0.83	12.33±0.87
SCAR-GAT	**7.93±0.13**	**8.12±0.53**	**5.94±0.75**

E.4 Visualization Analysis

In order to provide an intuitive and clear understanding of the effectiveness of the proposed model. We present visualizations of reliability diagrams and confidence distribution histograms with L/C = 20, across different datasets. The visualizations are shown in Figure 7.

In Figure 7 (Top), the first row represents the Reliability diagrams of GCN, and the second row represents the Reliability diagrams of the proposed SCAR. The x-axis divides the model's confidence into 20 equal intervals, while the y-axis represents the average accuracy for each interval. The gray area indicates the expected output values and the blue area represents the model's actual output values. The closer the bars are to the diagonal line (*i.e.,* $y = x$), the better calibrated the model. In all datasets, the uncalibrated GCN shows higher average accuracy than confidence in most bins, indicating under-confidence. Our proposed method effectively mitigates this under-confidence issue, aligning the model's confidence more closely with its accuracy.

In Figure 7 (Bottom), we visualize the confidence distribution of test nodes from different perspectives, where the x-axis represents the model's confidence and the y-axis represents the density at various confidence levels. The blue represents the confidence distribution of correct predictions, while the yellow represents that of incorrect predictions. From this visualization, we observe that in the case of an uncalibrated GCN, many correctly predicted samples fall into lower confidence intervals. After applying the proposed method, most correctly predicted samples shift to higher confidence intervals, indicating that the model assigns greater confidence to its correct predictions. This shift underscores the superiority of the proposed method in improving the reliability of predictions.

E.5 Analyzing the Trade-off Between Efficiency and Effectiveness

The proposed SCAR consists of two components. The first component involves a simple adjustment of the hyperparameters (i.e., class-centroid-level calibration), which does not introduce any additional complexity or runtime overhead. The second component, node-level calibration, modifies the

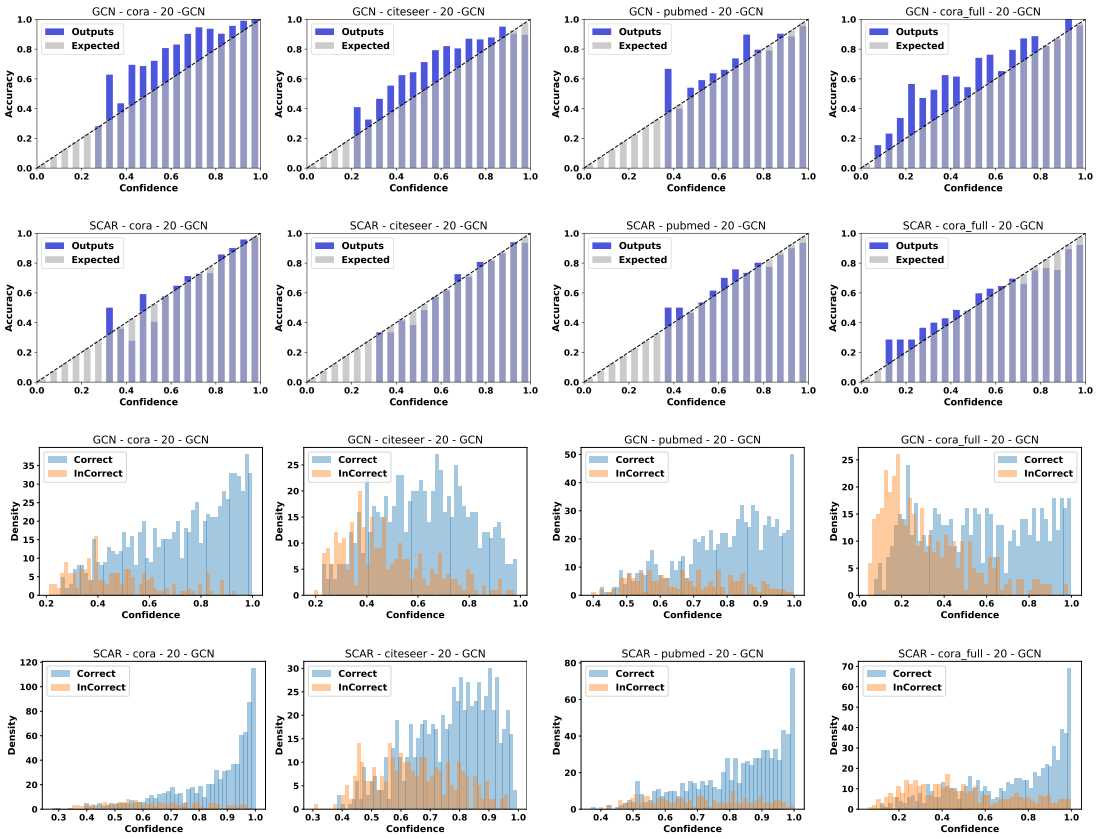


Figure 7: Visualization of Reliability diagrams (Top) and Confidence distribution histograms (Bottom) on Cora, Citeseer, Pubmed, and CoraFull datasets. The base model is GCN and L/C=20.

final-layer representation of the test nodes during model inference, as described in Eq. (10). The computational complexity of Eq. (10) is minimal, as the overhead caused by calculating the weighted sum once time is negligible. To evaluate the model's actual runtime performance, we present the running time and ECE of each model based on GCN across different datasets with L/C=20 in Figure 8. Note that a smaller ECE value indicates better calibration. Therefore, models positioned closer to the lower-left corner of the plot achieve a better trade-off between efficiency and effectiveness.

From Figure 8, we have the following observations: First, existing graph calibration methods fail to achieve a good trade-off between effectiveness and efficiency. For example, the trade-off results of the most effective baseline (*i.e.,* DCGC), on all datasets (represented by the pink points) are clustered in the lower-right corner. This indicates that, while DCGC achieves well-calibrated results, it incurs significant additional time overhead. In contrast, baselines with slightly worse ECE performance, such as GATS, demonstrate improvements in runtime efficiency. Second, the proposed SCAR (represented by the red points), on all datasets is clustered in the lower-left corner. In more detail, on the time axis, the proposed SCAR and its backbone model GCN are at the same level, indicating almost no time overhead generated by the proposed SCAR. On the ECE axis, the proposed method is unparalleled, achieving a distinct advantage over the baselines. This demonstrates that the proposed method has achieved the best trade-off between efficiency and effectiveness. This is attributed to the fact that the proposed calibration method is designed based on the mechanism of generating confidence within the GNNs. As a result, it relies solely on minor modifications to the model itself, without the need for any external components, thus introducing almost no additional time overhead.

E.6 Ablation Study on Node-Level Calibration

To further validate the refined node-level calibration in Eq. (10), we conduct an ablation study comparing (i) the full version that uses both parameters α and β, which explicitly accounts for the

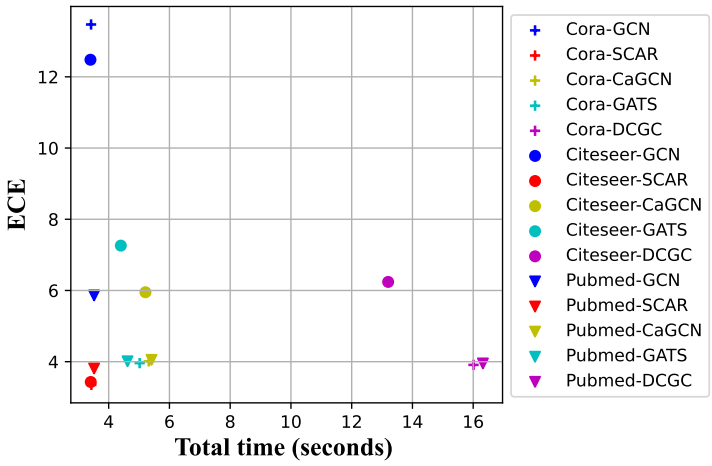


Figure 8: Scatter plot showing the relationship between model runtime and ECE, where the x-axis represents the runtime of different models on different datasets and the y-axis represents their corresponding ECE (%).

structural bias of GNNs, and (ii) a simplified variant that relies on a single parameter. Recall that the structural bias refers to the tendency of nodes closer to the training data in the graph to obtain representations that are more aligned with the class prototypes, and thus benefit differently from node-level adjustments.

Table 10 reports the calibration results across multiple datasets. The full version consistently achieves lower ECE and better confidence–accuracy alignment than the single-parameter variant. This indicates that modeling the structural bias through separate parameters for training-near and training-distant nodes leads to more appropriate adjustment strengths, thereby yielding improved calibration performance. These results confirm the necessity and effectiveness of the proposed refinement.

Table 10: ECE (%) of each component on all datasets with L/C=20. The best results are highlighted in black.

Method	GCN				GAT			
	Cora	**Citeseer**	**Pubmed**	**CoraFull**	**Cora**	**Citeseer**	**Pubmed**	**CoraFull**
Only+α	3.81±0.61	3.91±0.64	3.94±0.51	7.21±0.52	3.94±0.71	4.88±0.78	4.08±0.72	6.73±0.57
$+\alpha+\beta$	**3.35±0.65**	**3.43±0.58**	**3.81±0.53**	**6.96±0.48**	**3.52±0.74**	**4.37±0.83**	**3.78±0.84**	**6.41±0.63**

E.7 Evaluation on the Out-Of-Distribution Condition.

Table 11: ECE (%) of all methods on all datasets with L/C = 20, using GCN as the base model. Best results are highlighted in black. "ID" and "OOD" denote in-distribution and out-of-distribution, respectively.

Method	Cora		Citeseer	
	ID	**OOD**	**ID**	**OOD**
Uncal.	13.47±0.63	7.15±0.32	12.48±0.71	10.01±1.38
AU-LS	4.32±0.23	6.75±0.41	6.69±1.76	8.71±1.54
SCAR	**3.35±0.65**	**4.73±0.58**	**3.43±0.58**	**5.30±0.91**

To further examine the behavior of our method beyond the in-distribution (ID) setting, we conduct additional experiments on OOD graphs. Following the protocol of **(author?)** [48], we adopt Cora-OOD and Citeseer-OOD, where the original citation graph is treated as the in-distribution graph and

an OOD graph is generated via a stochastic block model (SBM). Specifically, the SBM graph is sampled with the same number of nodes as the original graph, and node labels are preserved while the edges are randomly rewired according to block-level connectivity probabilities. This setup creates a controlled distribution shift that affects both the graph topology and the smoothness of the label signal.

We evaluate the calibration performance of our method under this OOD setting and compare it with AU-LS, a recent post-hoc calibration baseline designed for graph data. The results are summarized in Table 11. From the Table 11, we have the following observation:

First, the uncalibrated models appear to show improved calibration on OOD graphs. This, however, is largely an artifact: the severe drop in accuracy caused by distribution shift makes the overall accuracy level closer to the confidence scores of under-confident GNNs. Such an apparent "improvement" does not reflect healthy calibration behavior, as it is achieved at the cost of dramatically degraded accuracy. Second, our method consistently improves calibration even under OOD conditions and achieves noticeable gains over AU-LS. This indicates that the proposed approach remains effective under moderate distribution shifts, although the improvement is understandably smaller than in the in-distribution setting.

F Reproducibility Statement

We have made extensive efforts to ensure the reproducibility of our work. The model architecture, training procedure, and hyperparameter settings are described in Appendix D.3. Complete proofs of the theoretical results are presented in Appendix B. The datasets used in our experiments are publicly available, and the preprocessing steps are explained in the Appendix D. Moreover, we provide a link to the source code in the supplementary materials to facilitate reproduction of our experiments (`https://github.com/huangJC0429/SCAR`).

G Broader Impacts

This paper proposes a novel calibration framework for GNNs. For the theoretical analysis of the final-layer weight decay and parameters, providing more explanations and motivations for researching the relationship between the model and confidence in the trustworthy machine learning field. Moreover, the proposed node-level calibration breaks the current reliance on temperature scaling as the sole foundation for graph post-hoc methods, introducing a new calibration framework for graph models. This can offer more choices for the following research and help design more flexible and effective calibration methods.

H Limitations and Future Works

While our work introduces an effective calibration framework for GNNs, it does not fully exploit structural biases, such as homophily and heterophily, which are known to affect prediction confidence. Incorporating these biases into the proposed calibration framework could further enhance its effectiveness, which we leave as an important direction for future work.

I LLM Usage Statement

In this work, a large language model (LLM) was used to polish the writing. The LLM assisted in improving clarity and grammar, but all scientific content, interpretations, and conclusions were generated solely by the authors.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are included in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the assumptions and complete proof in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 4 and Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We released codes and data at `https://anonymous.4open.science/r/SCAR-1066`

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Experiments are run on different seeds with the standard divination being reported in Tables 1, 2, 3, and 8.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We list the details of experiments compute resources in Appendix D.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix G.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Creators or original owners of code or data have been cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: In this work, a large language model (LLM) was used to polish the writing. The LLM assisted in improving clarity and grammar, but all scientific content, interpretations, and conclusions were generated solely by the authors.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.