Optimized Graph Structures for Calibrating Graph Neural Networks with Out-of-Distribution Nodes

Anonymous authors
Paper under double-blind review

Abstract

Graph neural networks (GNNs) achieve remarkable success in tasks such as node classification, link prediction, and graph classification. However, despite their effectiveness, the reliability of the GNN's prediction remains a major concern. particularly when the graphs contain out-of-distribution (OOD) nodes. Up to now, the calibration of GNNs in the presence of OOD nodes is still largely under-explored. Our empirical studies reveal that the calibration issue becomes significantly more complex when OOD nodes are present, and existing calibration methods prove to be less effective in this scenario. Recently, graph structure learning (GSL), a family of data-centric learning approaches, has proved to be effective in mitigating the adverse effects of the noisy information in the graph topology by optimizing the graph structure alongside with GNN training. However, current GSL methods do not explicitly address the calibration issue in graphs with OOD nodes. To tackle the this challenge, we propose a novel framework called Graph Calibration via Structure Optimization (GCSO) to calibrate GNNs against OOD nodes. Our empirical findings suggest that manually reducing the weight of edges connecting in-distribution (ID) nodes and OOD nodes could effectively mitigate the calibration issue. However, identifying these edges and determining their appropriate weights is challenging, as the distribution of OOD nodes is unknown. To address it, we propose a novel framework to calibrate GNNs against OOD nodes. In our method we first develop an iterative edge-sampling mechanism to capture the topological information of the graph and formulate it as the Markov Decision Process (MDP). Then, we leverage the actor-critic method to dynamically adjust the edge weights and assess their impact on target nodes. Additionally, we design a specialized reward signal to guide the policy function toward an optimal graph structure that minimizes the negative influence of OOD nodes. Note that our modified graph structure could be seamlessly integrated with existing temperature scaling-based calibration techniques for further improvement. Experimental results on benchmark datasets demonstrate that our method can effectively reduce the expected calibration error (ECE) while maintaining comparable accuracy in GNNs. And our approach outperforms strong baseline methods, The anonymous GitHub repository for the code is available at `https://anonymous.4open.science/r/calibration-7F61`.

1 Introduction

Graph neural networks (GNNs) have proven to be an effective approach to handle the graph-structured data, which is prevalent in our real world applications such as social networks, traffic networks and financial networks. In spite of their significant success in the tasks like node classification, link prediction and graph classification, the reliability of GNNs has become a growing concern in the machine learning community. A previous study (Guo et al., 2017) introduced the expected calibration error (ECE) to quantify the discrepancy between a model's predictive confidence and the actual likelihood of correctness. More recent works (Wang et al., 2021; Hsu et al., 2022; Teixeira et al., 2019; Fang et al., 2024; Yang et al., 2024b) pointed out that the GNNs often exhibit under-confidence in their predictions.

Existing GNN calibration method (Wang et al., 2021; Hsu et al., 2022; Fang et al., 2024; Tang et al., 2024) primarily focus on the "clean graph". However, real-world graphs often consist of out-of-distribution (OOD)

nodes (Zhao et al., 2020; Yang et al., 2022; Song & Wang, 2022). For instance, users on social networks are usually linked with strangers and online scammers apart from their family members and friends. When a graph is mixed with OOD nodes, the miscalibration of GNN tends to worsen.As illustrated in Fig 1, we designate nodes from specific classes (e.g., last two classes in Cora (Yang et al., 2016)) as OOD nodes, while treating the remaining nodes as in-distribution (ID) nodes. We then perform node classification using GCN (Kipf & Welling, 2016) on several graph benchmark datasets, and compare the ECE in scenarios with and without OOD nodes. The results indicate that presence of OOD nodes leads to an increase in ECE. Besides, the calibration issue becomes more complicated in the OOD scenario. As shown in Fig 2, unlike the general under-confidence problem of GNNs (Wang et al., 2021; Hsu et al., 2022) on clean graphs, In OOD scenarios, GNNs tend to be over-confident on some nodes while exhibiting under-confidence on others. Our experiments also show that the existing graph calibration methods would be less effective due to the presence of OOD nodes.

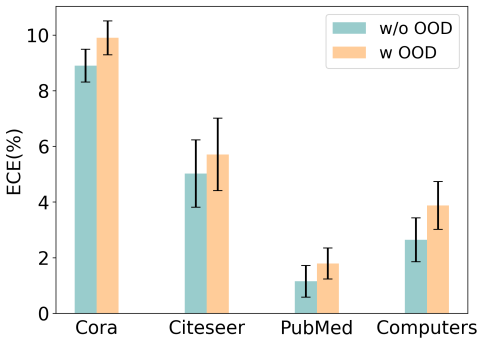


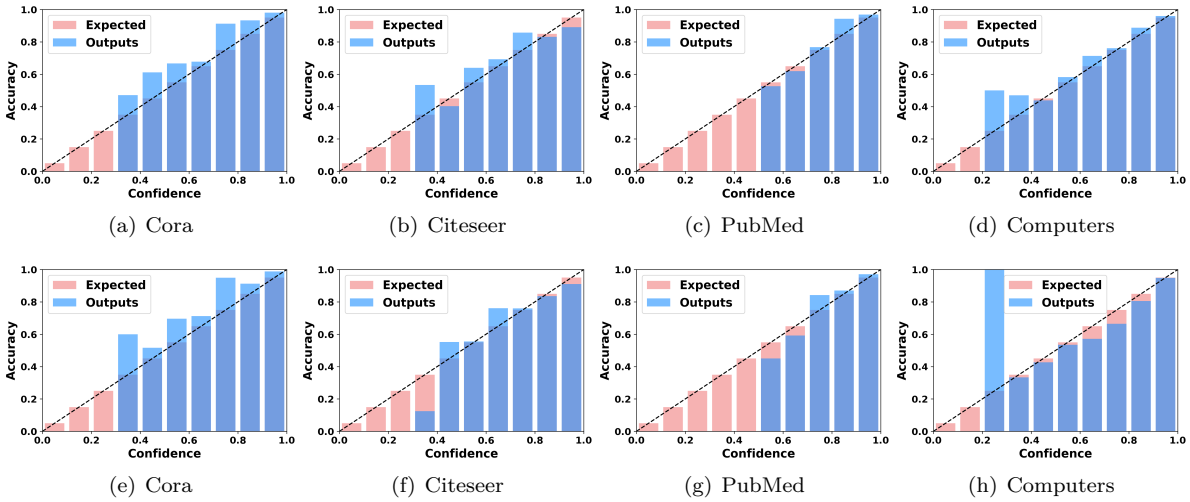
Figure 1: The expected calibration error (ECE) of GCN on graphs with and without OOD nodes.

Recently, graph structuring learning (GSL) (Wu et al., 2022; Zou et al., 2023) has exhibited promising results in mitigating the adverse effects of potential flaws, such as redundant, incorrect or missing connections, by optimizing the graph structure. Recently, Yang et al. (Yang et al., 2024a) proposed the Data-centric Graph Calibration (DCGC) framework to lower the calibration error by modifying the graph structure and assigning larger weights to decisive and homophilic edges. However, this work didn't explicitly account for OOD scenario. Shi et al. (Shi et al., 2023) leverage deep Q-learning (Mnih et al., 2013) to calibrate the graph with OOD nodes. However, their method assigned the fixed weight to the modified edges without considering the variations in the topological structure. For instance, edge weights should ideally adapt based on the distribution of OOD nodes. This limitation can lead to suboptimal calibration performance.

Inspired by the previous works (Yang et al., 2024a; Shi et al., 2023) that improve the calibration by reweighting the edges. We conduct an empirical study, and the results reveal that the calibration issue can be mitigated to some extent by adjusting the weights of edges that are connecting to OOD nodes. However, identifying the OOD nodes in the graph is not a trivial problem. To this end, we propose Graph Calibration via Structure Optimization (GCSO) framework to calibrate GNNs with the optimized graph structure. Our method follows the Actor-Critic paradigm. In our method, we select the labeled nodes as the target nodes and sample their adjacent edges. Then we introduce a novel iterative approach for these edges based on our predefined discrepancy score and formulate the iteration process as a Markov Decision Process (MDP). Next, we leverage the Actor-Critic paradigm to dynamically assess the impact of the adjusted edge weights on the target nodes. In particular, we adopt the deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016) in our framework to generate the fine-grained, topology-aware edge weights. In addition, a new reward signal is designed to guide the optimization of edge weight. The reward signal consists of two components: an indicator function and the entropy regularization term for the target nodes. The indicator function ensures accurate node classification with the optimized graph structure, while the entropy regularizes the logit distribution of the target nodes for the calibration purpose. Notably, our optimized graph structure can be seamlessly integrated with the existing post-hoc calibration method to further improve the calibration performance in the downstream tasks. The contribution of this paper is summarized as follows:

- We propose Graph Calibration via Structure Optimization (GCSO) to enhance the calibration of graph neural networks in the presence of OOD nodes. Our approach introduces a novel edge iteration method which is formulated as the Markov Decision Process (MDP). Additionally, A new reward signal is designed to guide the policy function to yield the optimized graph structure.

- Our method can be seamlessly integrated with the existing post-hoc calibration methods. Extensive experiments demonstrate that our method can outperform the baseline methods, achieving promising performance in the calibration of the GNNs when the graph contains OOD nodes.



(a) Cora (b) Citeseer (c) PubMed (d) Computers

(e) Cora (f) Citeseer (g) PubMed (h) Computers

Figure 2: Reliability diagrams of GCN on without OOD nodes ((a)-(d)) and with OOD nodes ((e)-(h)). The results suggest that the calibration issue (i.e., over-confidence or under-confidence) is more complicated with the presence of OOD nodes. Ideally, in a well-calibrated network, accuracy should align with confidence, meaning the height of the blue bars (accuracy) should be as close as possible to the height of the red bars (confidence). When the blue bar is taller than the red bar, it indicates under-confidence. Conversely, if the blue bar is shorter than the red bar, it indicates over-confidence.

- Experimental results further reveal that the learned edge weights are transferable, offering benefits in graph learning across various GNN architectures. Specifically, our optimized graph structure can enhance the performance in tasks such as node classification and OOD detection.

2 Related Works

Neural Network Calibration. The pursuit of developing a reliable and trustworthy model has captured the attention of researchers, leading to its extension into the realm of graph neural networks. Guo et al. (Guo et al., 2017) first proposed the calibration error to measure the confidence of the results from deep neural networks. Extensive work (Mukhoti et al., 2020; Ghosh et al., 2022; Tao et al., 2023; Wang et al., 2022; 2024; Tang et al., 2024) has been done on the calibration of neural networks. Recent work (Wang et al., 2021) post-processed the logits of the GCN (Kipf & Welling, 2016) model to obtain the calibrated results. Uncertainty estimation (Lakshminarayanan et al., 2017; Malinin & Gales, 2018) also benefits the network calibration by modeling the probability distribution of the predicted labels. Wang et al. (Wang et al., 2022) proposed GCL loss to mitigate the under-confidence issue of GNNs in an end-to-end manner. Besides, GATS (Hsu et al., 2022) is designed to account for the influential factors that affect the calibration of GNN. Fang et al. (Fang et al., 2024) highlighted that the ability of GNNs to distinguish between correct and incorrect predictions is crucial for achieving well-calibrated outcomes. And they propose a simple yet effective approach known as a Discriminative Calibration model for GNNs .Tang et al. (Tang et al., 2024) provided a theoretical insight on the role of nodewise similarity on the calibration of the GNN and proposed a novel calibration framework that takes advantage of the similarity on both global and local levels. Yang et al. (Yang et al., 2024b) highlighted a key limitation in existing GNN calibration methods, which predominantly focus on the highest logit while ignoring the full spectrum of prediction probabilities. To address this issue, they proposed a novel framework called Balanced Calibrated Graph Neural Network (BCGNN) (Yang et al., 2024b). This framework aims to achieve balanced calibration between over-confidence and under-confidence in the GNN predictions, which is supported by the solid theoretical justification. Unlike post-hoc methods that adjust temperature parameters for calibration, Yang et al. (Yang et al., 2024a) attempt to address the calibration issue from the data centric perspective. And their method aims to lower the calibration error by assigning larger weights to decisive and homophilic edges. Shi et al. (Shi et al., 2023) also investigated the calibration issue of graphs with presence of OOD nodes and proposed a new framework called GERDQ. However, there

are fundamental differences between our work and GERDQ (Shi et al., 2023). First, we introduce a novel edge iteration approach to better capture the topological structure of the graph. Second, while GERDQ (Shi et al., 2023) also calibrates graphs by reweighting edges, it assigns a fixed value to the new edge weights without considering variations in the topological structure (e.g., the distribution of OOD nodes). In contrast, our method generates fine-grained, topology-aware edge weights that adapt to the graph's structure. Lastly, we design a new reward function to guide the generation of an optimized graph structure, ensuring both improved node classification accuracy and better calibration performance.

Graph Structure Learning. Graph Structure Learning (GSL) aims to address graphs with unreliable, low-quality, or noisy structures, such as redundant or incomplete edges, by learning an optimized topology. Up to now, extensive research has been conducted in this field. Wu et al. (Wu et al., 2022) introduced a kernelized Gumbel-Softmax operator for efficiently approximating discrete latent structures among data point and proposed a transformer-based model to learn the optimal topology from node features and labels. To address the lack of robustness and interpretability in existing GSL methods, Zou et al. (Zou et al., 2023) proposed the SE-GSL framework, which can explicitly interpret the hierarchical semantics of graphs, and enhance the robustness of mainstream GNN approaches against noisy and heterophilous structures. To reduce the dependence of GSL methods on label information, Liu et al. (Liu et al., 2022) introduced Structure Bootstrapping contrastive Learning Framework (SUBLIME), a novel unsupervised learning framework that leverages self-supervised contrastive learning to optimize graph structure.

Reinforcement Learning on Graph. The rapid development of Reinforcement Learning (RL) in cross-disciplinary domains has motivated scholars to explore novel RL models to address graph-related problems, such as neighborhood detection, information aggregation, and adversarial attacks. GraphNAS (Gao et al., 2019) designs a search space covering sampling functions, aggregation functions, gated functions and searches the graph neural architectures with RL. Policy-GNN (Lai et al., 2020) adaptively determines the number of aggregations for each node via deep Q-learning (Mnih et al., 2013). RL-Explainer (Shan et al., 2021) and GFlowExplainer (Li et al., 2023) adopt off-policy reinforcement learning methods for graph explanation.

Graph Learning with OOD. Most graph learning is built on the hypothesis that training and testing data are independent and identically distributed (I.I.D.). Song et al. (Song & Wang, 2022) first proposed graph learning with OOD nodes and developed OODGAT (Song & Wang, 2022) framework to perform both the node classification and OOD nodes detection. The core of the OODGAT (Song & Wang, 2022) is to identify the OOD nodes and reduce the connection between ID nodes and OOD nodes. Another line of work focuses on graph OOD detection. GNNSAGE (Wu et al., 2023) performs OOD node detection by a learning-free energy belief propagation scheme. In GPN (Stadler et al., 2021) OOD nodes detection is completed by the uncertainty estimation. GraphDE (Li et al., 2022), a probabilistic generative framework, can jointly perform graph debiased learning and out-of-distribution nodes detection.

3 Preliminary

3.1 Problem Formulation

We first present the problem formulation of our study. Consider an attributed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ where the finite node set is denoted by $V = \{i | 1 \leq i \leq N\}$, and the edge set is denoted by $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. N is the total number of the nodes in the graph, and the feature matrix is denoted by $\mathbf{X} \in \mathbb{R}^{N \times d}$ in which d is the length of the feature vector. The structure of the graph $\mathcal{G}$ can be represented by the binary adjacency matrix $\mathbf{A} = \{0, 1\}^{N \times N}$. In graph learning with out-of-distribution (OOD) nodes, The nodes set can be split into an ID node set and an OOD node set $\mathcal{V} = \mathcal{V}_{ID} \cup \mathcal{V}_{OOD}$. The feature of OOD nodes is sampled from a different distribution than that of ID nodes, i.e., $P(X_{OOD}) \neq P(X_{ID})$. The label space for the ID node set is $Y = \{1, 2, \cdots, K\}$, while we assume that the OOD nodes do not fall into any existing category of the ID nodes, and their labels are unknown to us. In semi-supervised graph learning, the ID nodes can be further divided into labeled ID nodes and unlabeled ID nodes, i.e., $\mathcal{V}_{ID} = \mathcal{V}_{ID}^{l} \cup \mathcal{V}_{ID}^{ul}$. The goal of standard semi-supervised graph learning is to learn a classifier $f : \mathbf{X}, \mathbf{A} \rightarrow \tilde{Y}$ that maps the feature of the nodes and graph structural information to the predicted labels $\tilde{Y}$ of the nodes. As aforementioned, the task becomes

Table 1: Comparison between GCN with original and modified edge weights in terms of node classification accuracy (Acc%)↑ and expected calibration error (ECE%)↓. The experiments are repeated 10 times and the average results are reported. The bold represents the best results.

Edge weight	Cora		Citeseer		PubMed		CS		Computers		Arxiv	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
Original	84.18	9.90	71.57	5.41	92.11	1.59	**92.80**	2.93	90.81	3.88	42.47	6.14
Modified	**84.50**	**9.14**	**71.75**	**4.98**	**92.24**	**1.27**	92.68	**2.73**	**91.07**	**3.42**	**42.93**	**5.35**

more challenging with the presence of unknown OOD nodes. How to rule out the negative impact from the OOD nodes is the key for the semi-supervised graph learning with OOD nodes.

In our study, the expected calibration error (ECE) is considered as a major metric. According to the practice in related work (Guo et al., 2017), the predictions are regrouped into M equally spaced confidence intervals $(B_1, B_2, \cdots, B_M)$ with $B_m = \{i \in \mathcal{V} | \frac{m-1}{M} < \tilde{p}_i \leq \frac{m}{M}\}$ where $\tilde{p}_i$ is the confidence for node i. And the expected calibrated error (ECE) can be defined as $\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{|\mathcal{V}|} |\text{acc}(B_m) - \text{conf}(B_m)|$, where $\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\tilde{y}_i = y_i)$ and $\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \tilde{p}_i$.

3.2 Deep Reinforcement Learning

Reinforcement learning plays an important role in the decision making process, and one representative method is the Markov Decision Process (MDP). A typical MDP can be formulated as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P_\pi, r, \gamma\}, \rho_0$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P_\pi(s'|s, a)$ is the state-action transition probability, r is the reward function, r is the reward function, $\gamma \in (0, 1)$ is the discount factor ,and ρ_0 is the initial state distribution over state space $\mathcal{S}$. The goal of off-policy reinforcement learning is to learn the policy $\pi(a|s)$ that can maximize the discounted cumulative reward $J_\pi = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ by training on the outcomes produced by a different behavior policy rather than that produced by the target policy. One of the most well-known off-policy method in deep learning is deep Q-learning (Mnih et al., 2013; Van Hasselt et al., 2016). The basic idea of deep Q-learning is to approximate the Q function by deep neural networks, and the policy is obtained from the estimated value of $a = \text{argmax}_a Q(s, a) = \text{argmax}_a \mathbb{E}_{s' \sim \mathcal{S}}(r + \gamma \max_{a'} Q(s', a'))$.

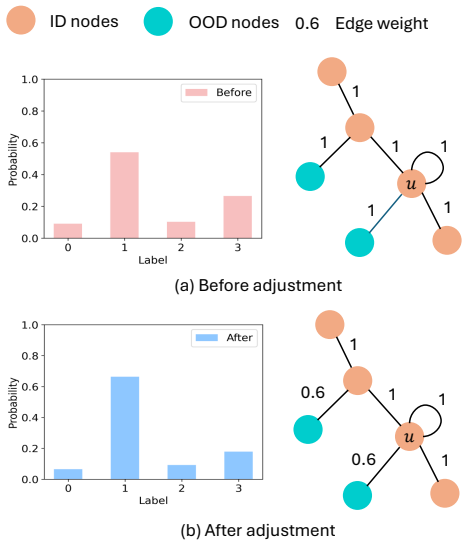


Figure 3: The change of logit distribution before and after adjustment of edge weight. The result is yielded by GCN on Cora.

Apart from Q-value based methods that obtain the action implicitly from the Q function, policy gradient methods (Haarnoja et al., 2018; Wang et al., 2017; Cobbe et al., 2021; Barth-Maron et al., 2018; Tkachenko, 2015; Silver et al., 2014b; Mnih et al., 2016) instead aim to learn the policy directly by parameterized function $\pi_\theta(a)$. Similar to deep Q-learning (Mnih et al., 2013; Van Hasselt et al., 2016), we update the parameter θ in the policy function to achieve the maximum discounted cumulative reward. Besides, modern off-policy gradient methods (Haarnoja et al., 2018; Wang et al., 2017; Cobbe et al., 2021; Barth-Maron et al., 2018; Tkachenko, 2015) adopt the actor-critic algorithm that models the policy and Q function to achieve better learning efficiency and convergence. The parameter θ of policy function can be updated according to the Policy Gradient Theorem (Sutton et al., 1999):

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[\nabla \ln \pi(a|s, \theta) Q_\pi(s, a)]. \tag{1}$$

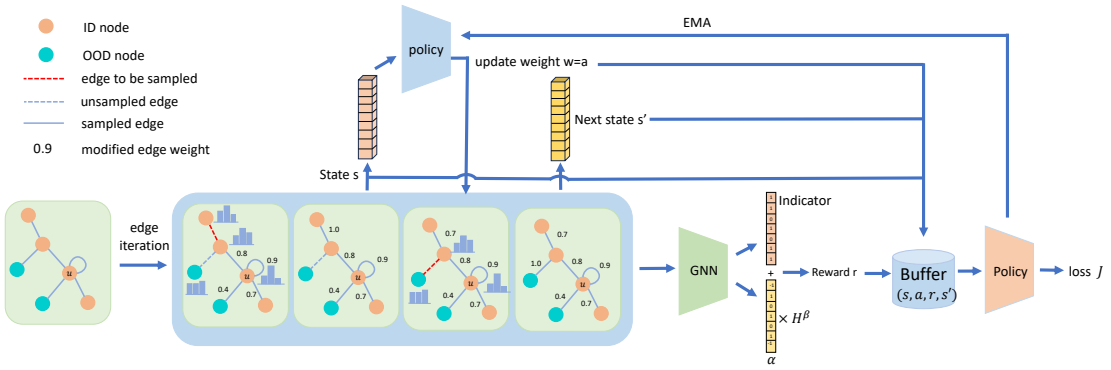


Figure 4: The illustration of our proposed Graph Calibration via Structure Optimization framework. The method consists of four steps. First, we iteratively traverse the adjacent edges from the candidate edge set. In the beginning, only the self-loop edge is taken into consideration. Each time we sample a new edge within the subgraph without replacement according to the discrepancy score and form the state. Second, the adjusted weight would be obtained from the state and assigned to the new sampled edge. Next, reward r is obtained from the GNN with adjusted edge weight and a new state is also formed. Next, the transition tuple is stored in the replay buffer. Finally, we adopt the DDPG method to train our policy function.

4 Empirical Study

In this section we investigate if the calibration error of GNNs can be reduced with adjusted edge weights when a graph is mixed with OOD nodes, We follow the previous work (Zhao et al., 2020; Stadler et al., 2021) to divide the the existing nodes into ID nodes and OOD nodes and choose GCN (Kipf & Welling, 2016) as the target model. Suppose the labels and distribution of all nodes is known and we manually modify the weight of edges (e.g., 1 to 0.6) that are connecting to OOD nodes. The experiments are evaluated on the six benchmark datasets. The details of the benchmark can be found in Table 2. The results in Table 1 suggest that lowering the weight of corresponding edges can definitely reduce the calibration error while maintaining comparable accuracy on node classification compared to that with original edge weights. The intuition is that the modified edge weight could regularize the entropy of the node so that the predictive confidence would be changed. To validate our hypothesis we obtain the output distribution of GCN (Kipf & Welling, 2016) on Cora (Yang et al., 2016) before and after the edge weight adjustment. As shown in Fig 3, the results suggest that the predictive confidence is slightly changed while the prediction is preserved. Based on these findings, we are motivated to develop new methods to obtain new edge weights to calibrate GNNs.

5 Methodology

In this section, we give an overview of our framework. In this section, we first introduce the formulation of our edge iteration process and the definition of the key elements in our method such as State, Action and Reward. Then we provide details of our training pipeline. Besides, we provide a discussion of our method along with an analysis of the time complexity.

5.1 Iterative Edge Sampling and Re-weighting

We first form the candidate nodes set $\mathcal{I}$ which are sampled from node set $\mathcal{V}$. Then the candidate edges for iteration are sampled from adjacent edges of a target node $u \in \mathcal{I}$ and the edge set is denoted as $\mathcal{E}^u = \{e_0^u, e_1^u, \cdots, e_{k-1}^u\}$. Specifically, e_0^u is the self-loop edge for node u. During the iteration an edge is sampled from the candidate edge set and the weight is adjusted accordingly. Specifically, At time $t = 0$, we only consider the re-weighting of the self-loop edge e_0^u. From time $t-1$ to t, a new edge is chosen from $\mathcal{E}^u$. In our framework the iterative edge sampling and the re-weighting process are formulated as a Markov Decision Process (MDP) and the definitions of state, action, and reward are illustrated as follows.

State. The state $s_t \in \mathcal{S}$ at timestamp t in our framework is defined as:

$$s_t = h(s_{t-1}, f_{e_t}), \tag{2}$$

where f_e is the feature of the edge e, and h is the function that maps the old state and new edge feature into the new state. We adopt the average of the features of the connecting nodes of the edge e as the edge feature. At time $t = 0$, $s_0 = X_u$. In our study, the moving average method is adopted to generate the state. The state at time t can be formulated as:

$$s_t = \lambda f_{e_t} + (1 - \lambda)s_{t-1}, \tag{3}$$

where λ is the hyper-parameter that balances the contribution of new edge features in the state.

In each iteration, a new edge is chosen from the candidate edge set $\mathcal{E}^u$ according to the discrepancy score δ. The discrepancy score δ measures the correlation between an adjacent edge e^u and the target node u. It is defined as:

$$\delta := \frac{1}{2}(\mathrm{KL}(z_1||z_u) + \mathrm{KL}(z_2||z_u)), \tag{4}$$

where z_1, z_2 and z_u are the logit distribution of the connecting nodes of the edge e^u and target node u, respectively. KL denotes KL divergence. Intuitively, the discrepancy score would be larger if an edge is connecting to an OOD node of which the logit distribution would deviate from that of ID nodes. In each iteration, when the policy function yields new edge weights, the logit distribution of the nodes would be updated and the edge with lowest discrepancy score would be chosen from the candidate edge set. Note the edge selection is a sampling without replacement process and only the discrepancy score of the unsampled edges would be compared. The motivation behind this selection is straightforward. Since node similarity plays an important role in the calibration of the nodes, edge sampling with discrepancy score would be beneficial to discern the effect of nodes with different similarity on the calibration of the target node and facilitate the convergence of the training.

Action. In our method, the action $a \in \mathcal{A}$ we take for each new sampled edge is to adjust its weight. Since in our case, the action space is continuous $\mathcal{A} \subseteq (0, 1]$, we adopt the policy function to generate the adjusted edge weight from the state s. At time t, the edge weight w_{e_t} for e_t is generated by:

$$w_{e_t} = \pi(s_t|\theta^\pi), \tag{5}$$

where π_θ is the policy function which can be implemented as a neural network with the Sigmoid activation function in the last layer to ensure the output is between 0 and 1.

Reward. The reward signal r is designed to encourage the policy function to produce new edge weights to regularize the logit distribution of the target nodes. To determine if the node is over-confident or under-confident, we evaluate the calibration error on the validation nodes and obtain the $\mathrm{acc}(B_m)$ and $\mathrm{conf}(B_m)$ for each bin during training. If the predictive probability of the target node falls into certain bin m, then the reward would be defined as

$$r(s, a) = \mathbb{1}(\tilde{y}_i = y_i) + \alpha H_i^\beta, \tag{6}$$

where

$$\begin{cases} \alpha = +1, \beta = 1 & \text{if } \tilde{y}_i = y_i \text{ and } \mathrm{acc}(B_m) < \mathrm{conf}(B_m) \\ \alpha = -1, \beta = 1 & \text{if } \tilde{y}_i = y_i \text{ and } \mathrm{acc}(B_m) > \mathrm{conf}(B_m) \\ \alpha = 0, \beta = 0 & \text{if } \tilde{y}_i = y_i \text{ and } \mathrm{acc}(B_m) = \mathrm{conf}(B_m) \\ \alpha = 1, \beta = 1 & \text{if } \tilde{y}_i \neq y_i \end{cases} \tag{7}$$

$\tilde{y}_i$ is the predicted label for node i generated by the GNN backbone, and y_i is the ground truth label. H denotes the entropy of the target node u.

5.2 Details of Algorithm

The framework of our proposed method is illustrated in Fig. 4. The framework basically consists of four steps. In this first step, we form the candidate node set $\mathcal{I}$ from the training and validation nodes. For each

candidate node, we iteratively sample the adjacent edges and form the state, as discussed in Sec. 5.1. In step two, the adjusted edge weight is obtained from the policy function $\pi_\theta(s)$. In order to enhance the exploration ability of the policy function in the continuous action space, we reformulate our adjusted edge weight as:

$$w^*_{e_t} = \pi(s_t|\theta^\pi) + \epsilon, \tag{8}$$

where ϵ is the noise following a uniform distribution and the upper bound is determined by $\epsilon_0(1 + \frac{t}{T})^{-d}$, where ϵ_0 is the initial noise. T is the total number of iterations. $d > 0$ is the decay rate. In the next step we obtain the reward r from the GNN backbone according to Eq. equation 6 and the tuple of transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer $\mathcal{B}$. In the final step, we adopt the deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016) method to train our policy function, because the state and action spaces are all continuous in our problem. DDPG (Lillicrap et al., 2016) adopts the actor-critic framework for better stability and convergence of the training. Similar to deep Q-learning (Mnih et al., 2013), the objective of critic network $Q(s_t, a_t|\theta^Q)$ is to approximate the discounted cumulative reward from the state-action pair by minimizing the loss:

$$L(\theta^Q) = \mathbb{E}_{s_t\sim\mathcal{S}, a_t\sim\mathcal{A}}[(Q(s_t, a_t)|\theta^Q] - y_t)^2], \tag{9}$$

where y_t can be derived from the Bellman equation (Sutton & Barto, 2018) $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}|\theta^\pi)|\theta^Q)$. Since our policy function yields the continuous edge weight deterministically from the state, the parameter of policy can be updated according to the Deterministic Policy Gradient Theorem (Silver et al., 2014a; Lillicrap et al., 2016):

$$\begin{aligned}\nabla_{\theta^\pi} J &= \mathbb{E}_{s_t}[\nabla_a Q(s, a|\theta^Q)_{s=s_t, a=\pi(s_t|\theta^\pi)} \nabla_{\theta^\pi}\pi(s|\theta^\pi)_{s=s_t}] \\ &\approx \frac{1}{N}\sum_i (\nabla_a Q(s, a|\theta^Q)_{s=s_i, a=\pi(s_i|\theta^\pi)} \nabla_{\theta^\pi}\pi(s|\theta^\pi)_{s=s_i}).\end{aligned} \tag{10}$$

The detailed procedures of our proposed method are summarized in Algorithm 1 in the Appendix.

5.3 Time Complexity

Our framework consists of a GNN and actor/critic network. Suppose the L is the number of layers in GCN, $|E|$ is the total number of edges, N is the total number of nodes, d is the dimension of the features, $|\mathcal{E}^u|$ is the number of edges in the candidate edge set. $|\mathcal{I}|$ is the number of the target nodes. The time complexity for GNN and actor/critic networks are $O(L|E||\mathcal{E}^u|d + LN|\mathcal{E}^u|d^2)$ and $O(|\mathcal{I}||\mathcal{E}^u|d)$, respectively. Thus, the total time complexity is $O(|\mathcal{E}^u|(L|E| + |\mathcal{I}|)d + LN|\mathcal{E}^u|d^2)$. The major factors that influence computational efficiency in our approach are the number of target nodes and the number of adjacent edges for iteration. As the graph size increases, the number of nodes and adjacent edges also grows, raising the computational cost of our method. To address this issue, we limit the number of target nodes and adjacent edges for each iteration. For larger graphs like OGB-Arxiv (Hu et al., 2020), we select up to 48 target nodes and 64 adjacent edges per node. Additionally, to further improve computational efficiency, we process edge iterations for all target nodes in batches. As a result, our method is scalable to large graphs and it can maintain a reasonable computational cost even as graph size increases. During the inference time, the modified edge weights would be yielded by our lightweight policy function in batch. Fig 5 illustrates the actual training time of the data-centric method DCGC (Yang et al., 2024a) and our proposed method on different benchmarks. The results suggest that, although our method requires more training time than the baseline, it maintains a reasonable computational cost even for large graphs.

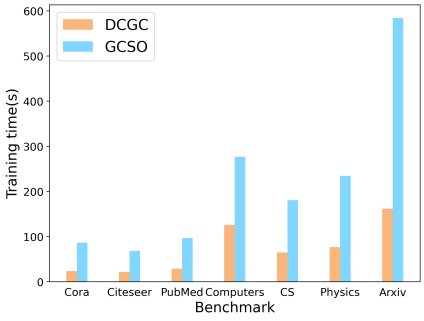


Figure 5: The actual training time of the baseline (DCGC) our method on different benchmarks. The evaluation is conducted on an NVIDIA RTX A5000.

Table 2: The statistics of datasets

Dataset	ID classes	OOD classes	#Nodes	#Edges	#Features
Cora	[0 - 4]	[5 - 6]	2,708	10,556	1,433
Citeseer	[0 - 3]	[4 - 5]	3,327	9,104	3,703
PubMed	[0 - 1]	[2]	19,717	88,648	500
Chameleon	[0 - 2]	[3 - 4]	2,277	36,101	2,325
Coauthor-CS	[0 - 10]	[11 - 14]	18,333	163,788	6,805
Coauthor-Physics	[0 - 2]	[3]	34,493	495,924	8,415
Amazon-Computers	[0 - 6]	[7 - 9]	13,752	491,722	767
OGB-Arxiv	[0 - 29]	[30 - 39]	169,343	1,166,243	128

5.4 Discussion

In our method, we design a specialized reward signal to refine the graph structure, which could implicitly regularize the logit distribution of the GNNs. When the target node is correctly predicted and the corresponding accuracy is lower than the confidence, the reward is $r = 1 + H$. An increasing reward enlarges the entropy, making the logit distribution less concentrated (e.g., reducing confidence). When the corresponding accuracy exceeds the confidence, the reward $r = 1 - H$ reduces the entropy, making the output distribution more concentrated (e.g., increasing confidence). If the target node is not correctly predicted. The reward $r = H$ simply makes the model deviate from the current output distribution.

Next, we discuss the generalization of our method. Our policy function is trained on selected edges within a graph. Since there is no distribution shift within a single graph, our method can generalize to other edges in the same graph. However, due to the differences in topological structures across graphs, the policy functions for different graphs need to be trained separately. Another advantage of our method is that the modified graph structure can be leveraged by other calibration frameworks (e.g., CaGCN (Wang et al., 2021) and GATS (Hsu et al., 2022)) to achieve better calibration results.

Besides, earlier work (Zhang et al., 2020) suggests three desired properties for calibration methods: **accuracy-preserving**, **data-efficient**, and **expressive**. Our method can fulfill these properties. First, our method adopts an indicator function in the reward signal to ensure the adjusted graph structure would not compromise the accuracy on the ID nodes. Besides, our method is also data-efficient. In our method we adopt GCN (Kipf & Welling, 2016) as the GNN backbone and lightweight MLPs for the actor and critic models, respectively. At last, our method is expressive, as it could produce a fine-grained, topology-aware weight for each edge.

Finally, our work aligns with a prior study (Shi et al., 2023) that also addresses the calibration issue of graph neural networks (GNNs) in out-of-distribution (OOD) scenarios. Although both works utilize reinforcement learning, there are significant differences between the two studies. First, the previous work lacks a solid empirical foundation and is driven by more intuitive motivations. In contrast, our research identifies the presence of both under-confidence and over-confidence in GNNs with OOD nodes, an observation not made in the prior study which led us to develop a different approach. Furthermore, compared to the previous work (Shi et al., 2023), we develop a new edge iteration process to capture the topological information of the graph. Furthermore, we propose a novel reward signal to generate fine-grained, topology-aware edge weights for the adjusted graph structure. In contrast, the previous work (Shi et al., 2023) produces fixed edge weights without considering topological information, such as the distribution of OOD nodes.

6 Experiments

In this section, we first introduce the experimental settings. Then we show the main results of the experiment as well as the visualization of the reliability diagrams and the distribution of the modified edge weights. The results of ablation study and case study can be found in Appendix.

Table 3: Comparison between our proposed method and other baselines in terms of node classification accuracy (Acc%)↑ and expected calibration error (ECE%)↓ on Cora, Citeseer and PubMed. The experiments are repeated 10 times and the average results and standard deviation are reported. Note that the primary focus of our study is the ECE performance of the methods.

Methods	Cora		Citeseer		PubMed		Chameleon	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
GCN (Kipf & Welling, 2016)	84.18 ± 0.28	9.90 ± 0.61	71.57 ± 0.73	5.41 ± 1.51	92.11 ± 0.17	1.59 ± 0.56	47.27 ± 1.27	14.81 ± 2.14
GCL (Wang et al., 2022)	84.19 ± 0.25	10.05 ± 0.63	71.91 ± 0.96	6.07 ± 2.03	92.14 ± 0.14	1.56 ± 0.25	45.81 ± 0.72	13.98 ± 1.79
OODGAT (Song & Wang, 2022)	83.17 ± 1.34	13.96 ± 3.87	61.95 ± 0.78	8.52 ± 2.08	87.44 ± 0.91	4.64 ± 1.28	39.74 ± 4.57	11.06 ± 5.90
HyperU-GCN (Yang et al., 2022)	81.88 ± 1.09	8.40 ± 7.75	71.27 ± 1.39	19.69 ± 13.74	92.35 ± 0.48	3.24 ± 0.77	47.36 ± 2.29	15.22 ± 9.65
CaGCN (Wang et al., 2021)	84.14 ± 0.35	3.85 ± 1.05	71.57 ± 0.73	4.27 ± 0.62	92.11 ± 0.17	3.09 ± 0.20	47.27 ± 1.27	14.15 ± 1.24
GATS (Hsu et al., 2022)	83.49 ± 0.31	2.81 ± 0.82	72.04 ± 0.46	5.05 ± 1.86	92.56 ± 0.24	2.27 ± 0.39	46.76 ± 2.73	11.18 ± 3.62
GERDQ (Shi et al., 2023)	83.67 ± 0.48	9.54 ± 0.50	69.98 ± 0.55	4.36 ± 0.92	92.14 ± 0.20	1.60 ± 0.59	46.87 ± 1.54	14.22 ± 1.65
DCGC (Yang et al., 2024a)	83.91 ± 0.25	10.44 ± 0.76	65.02 ± 0.65	4.62 ± 1.07	92.26 ± 0.16	2.43 ± 0.44	**47.60** ± 3.22	13.28 ± 4.42
GCSO (Ours)	**84.95** ± 0.18	9.22 ± 0.49	71.80 ± 0.70	4.55 ± 1.63	92.16 ± 0.16	**1.49** ± 0.20	46.81 ± 1.03	13.03 ± 1.59
GCSO+CaGCN (Ours)	84.28 ± 0.27	**2.55** ± 0.45	71.82 ± 0.68	**4.15** ± 0.48	92.24 ± 0.26	2.80 ± 0.19	47.28 ± 1.21	13.78 ± 0.82
GCSO+GATS (Ours)	84.20 ± 0.31	2.63 ± 0.46	**72.24** ± 0.90	4.20 ± 0.48	**92.69** ± 0.27	2.13 ± 0.34	46.80 ± 1.95	**10.07** ± 2.07

Table 4: Comparison between our proposed method and other baselines on Photo, Computers and Arxiv in terms of node classification accuracy (Acc%)↑ and expected calibration error (ECE%)↓. The experiments are repeated 10 times and the average results and standard deviation are reported. Note that the primary focus of our study is the ECE performance of the methods.

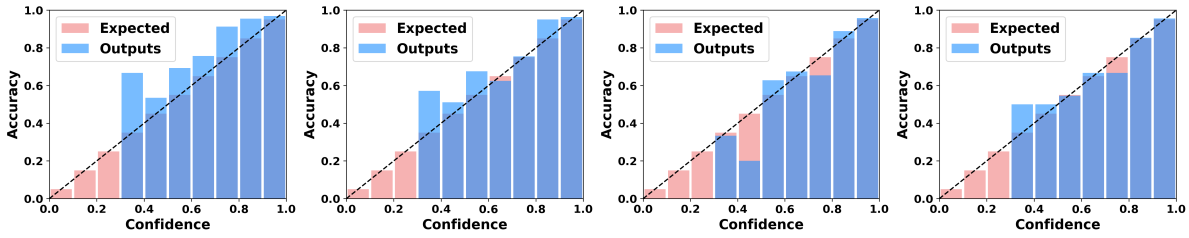
Methods	Coauthor-CS		Coauthor-Physics		Amazon-Computers		OGB-Arxiv	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
GCN (Kipf & Welling, 2016)	92.80 ± 0.45	2.93 ± 0.30	97.02 ± 0.21	1.27 ± 0.14	90.81 ± 0.53	3.88 ± 0.86	42.47 ± 0.67	6.14 ± 0.79
GCL (Wang et al., 2022)	92.83 ± 0.41	2.91 ± 0.29	97.04 ± 0.21	1.31 ± 0.12	90.95 ± 1.04	3.40 ± 0.59	42.53 ± 0.63	6.41 ± 0.79
OODGAT (Song & Wang, 2022)	90.63 ± 0.35	4.16 ± 0.60	93.79 ± 0.52	3.44 ± 0.38	90.89 ± 1.02	4.84 ± 1.01	42.11 ± 1.19	11.64 ± 0.74
HyperU-GCN (Yang et al., 2022)	90.74 ± 1.03	2.94 ± 0.84	96.37 ± 0.57	1.86 ± 0.70	91.17 ± 1.37	5.82 ± 1.10	36.72 ± 0.65	13.23 ± 1.58
CaGCN (Wang et al., 2021)	92.80 ± 0.44	3.43 ± 0.39	97.04 ± 0.20	1.09 ± 0.13	90.89 ± 0.76	2.64 ± 0.40	41.99 ± 0.75	4.42 ± 0.35
GATS (Hsu et al., 2022)	92.17 ± 0.61	3.22 ± 0.42	96.80 ± 0.34	1.21 ± 0.21	90.75 ± 1.00	3.67 ± 0.42	42.07 ± 0.79	4.84 ± 0.36
GERDQ (Shi et al., 2023)	92.82 ± 0.43	2.88 ± 0.29	97.05 ± 0.23	1.38 ± 0.23	91.16 ± 0.67	3.64 ± 0.85	43.58 ± 0.60	4.70 ± 0.48
DCGC (Yang et al., 2024a)	92.80 ± 0.26	2.92 ± 0.18	96.97 ± 0.17	1.52 ± 0.17	91.17 ± 0.60	3.36 ± 0.24	43.62 ± 0.54	4.85 ± 0.39
GCSO (Ours)	92.70 ± 0.47	**2.79** ± 0.33	**97.08** ± 0.21	1.24 ± 0.12	91.20 ± 0.48	3.56 ± 0.49	**43.64** ± 0.93	4.35 ± 0.49
GCSO+CaGCN (Ours)	**92.85** ± 0.47	3.33 ± 0.35	97.02 ± 0.21	**1.06** ± 0.06	**91.25** ± 0.49	**2.38** ± 0.16	42.59 ± 0.51	**4.16** ± 0.39
GCSO+GATS (Ours)	91.77 ± 0.17	3.09 ± 0.28	96.78 ± 0.33	1.15 ± 0.09	90.71 ± 0.57	3.52 ± 0.42	42.89 ± 0.71	4.37 ± 0.64

6.1 Experimental Settings

In the experiments, we perform the semi-supervised node classification task and compare the performance of our framework with the baseline methods on six benchmark datasets. The ablation study and case study can be found in Appendix.

Datasets. We adopt eight public benchmark datasets, including Cora, Citeseer, PubMed (Yang et al., 2016), Chameleon (Rozemberczki et al., 2021), Coauthor-CS, Coauthor-Physics, Amazon-Computers (Shchur et al., 2018) and OGB-Arxiv (Hu et al., 2020), for evaluating our method and baselines. Among the dataset, the Chameleon (Rozemberczki et al., 2021)is a heterophilous graph while others are homophilous graphs. We adhere to the train/validation/test splits provided by previous work (Yang et al., 2016; Shchur et al., 2018). To formulate the graph learning with the OOD nodes setting, we manually split the nodes into ID nodes and OOD nodes according to the routine from the previous work (Gal & Ghahramani, 2016; Song & Wang, 2022; Stadler et al., 2021). For instance, Cora (Yang et al., 2016) has 7 classes and the nodes from the last 2 classes would be regarded as OOD nodes which are marked out in the training and validation data. The rest would be ID nodes. More details of the datasets are illustrated in Table 2.

Baselines. The baselines include GCN (Kipf & Welling, 2016), HyperU-GCN (Yang et al., 2022), CaGCN (Wang et al., 2021), GATS (Hsu et al., 2022), GCL (Wang et al., 2022), OODGAT (Song & Wang, 2022), GERDQ (Shi et al., 2023) and DCGC (Yang et al., 2024a). More details about the baseline methods can be found in Appendix.



(a) GCN (b) HyperU-GCN (c) CaGCN (d) Ours

Figure 6: Reliability diagrams of different methods on Cora with OOD nodes. Well-calibrated results would have closer alignment with the expected results along the diagonal line. The results suggest that the calibration issue is different and complicated on different datasets.

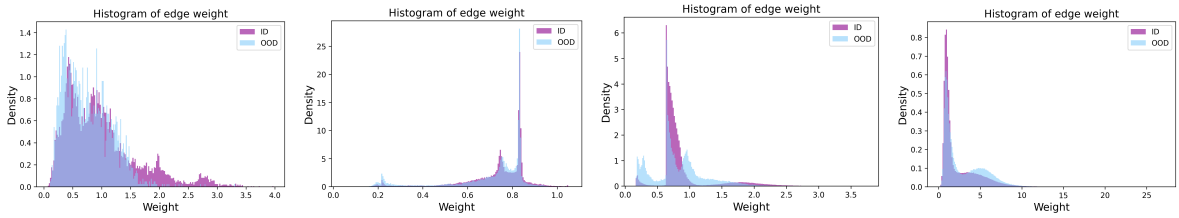
Metrics. In our experiments, we adopt the expected calibration error (ECE) (Guo et al., 2017) as our major metric. The lower value of ECE means the better reliability of the prediction results from GNN models. Besides, we also report the node classification accuracy.

Implementation Details. In our method, we adopt GCN (Kipf & Welling, 2016) as our GNN backbone. The hyper-parameters of GCN are the same as the corresponding baselines. The learning rate is 1e-2 and weight decay is 5e-4. The hidden dimension is 64. The Actor and Critic in our framework are implemented as a three-layered MLP with the dimension of hidden layers 256 and 16, respectively. More details about the implementation details can be found in Appendix.

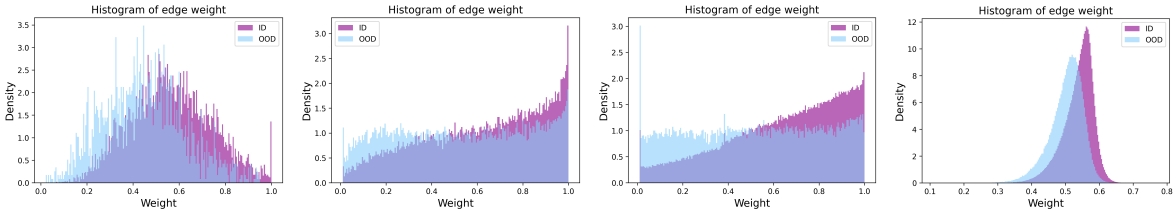
6.2 Main Results

Table 3 and Table 4 show the performance of our proposed method and the baselines on the benchmarks. The results show that the ordinary GNN models such as GCN (Kipf & Welling, 2016) would yield large calibration errors. For instance, the ECE can reach an average 9.90% on Cora (Yang et al., 2016). Besides, the results also suggest that the methods aimed at the calibration of GNNs can basically improve the calibration performance on GCN (Kipf & Welling, 2016). However, it could fail on some benchmark datasets. For instance, Although CaGCN (Wang et al., 2021) can achieve the low calibration error on Cora (Yang et al., 2016) and Amazon-Computers (Shchur et al., 2018), it still results in poorer calibration on PubMed (Yang et al., 2016) and Coauthor-CS (Shchur et al., 2018) than that of GCN (Kipf & Welling, 2016). The cause of the phenomenon can be attributed to the compromised homophily of the graph and make the regularization term in CaGCN (Shchur et al., 2018) less effective on these dataset. GCL (Wang et al., 2022) is also less effective on some datasets. OODGAT (Song & Wang, 2022) can identify the potential OOD nodes during the training and reduce the connection between the ID and OOD nodes by lowering the corresponding edge weights. However, our experimental results show that it would still suffer large calibration errors on some benchmark datasets. Additionally, our experimental results reveal that GERDQ (Shi et al., 2023) and DCGC (Yang et al., 2024a) effectively reduce the calibration error across various benchmarks. This further validates that refining the graph topology is beneficial for lowering the calibration error, particularly in graphs with the presence of OOD nodes.

The experimental results suggest that our proposed GCSO method is effective in calibrating GNNs and achieves better ECE performance compared to GERDQ (Shi et al., 2023) and DCGC (Yang et al., 2024a) on most datasets. DCGC (Yang et al., 2024a) can generate an adjusted graph structure to reduce the calibration error. However, this method does not account for the presence of OOD nodes. GERDQ (Shi et al., 2023) also employed adjusted weights to calibrate GNN results. However, its adjustment is guided solely by accuracy signals, which limits its calibration performance. In contrast, our approach employs a specialized reward signal to dynamically assess the impact of adjusted edge weights on target nodes, generating an optimal, topology-aware graph structure that regularizes the predictive confidence of GNNs. When integrated with existing graph calibration methods, our approach can further enhance calibration performance. For instance, GCSO+GCN can achieve the ECE of 2.55% and 2.38% on Cora (Yang et al., 2016) and Amazon-Computers (Shchur et al., 2018).



(a) Cora (b) PubMed (c) Physics (d) Arxiv



(e) Cora (f) PubMed (g) Physics (h) Arxiv

Figure 7: The distribution of edge weights connecting to ID nodes and OOD nodes from DCGC ((a)-(d)) and our proposed method ((e)-(h)) on various datasets. Compared to DCGC, the edge weights generated by our method exhibit a clear distribution shift between ID and OOD nodes.

6.3 Visualization

To better visualize the ECE, the reliability diagrams for our method and the baselines On Cora (Yang et al., 2016) are illustrated in Fig. 6. Well-calibrated results are supposed to have closer alignment with the expected results along the diagonal line. Fig. 6 demonstrates the better alignment of our method to the diagonal line than that of other baselines, which is consistent with our experimental results. The experiments suggest that both two terms are an indispensable part of the reward signal in our framework.

We also visualize the modified graph structures of both DCGC (Yang et al., 2024a) and our proposed method on various datasets. The histogram in Fig. 7 illustrates the distribution of edge weights connecting ID and OOD nodes, respectively. Since DCGC (Yang et al., 2024a) doesn't explicitly account for OOD nodes, the distribution of edge weights connecting to ID and OOD nodes shows significant overlap. In contrast, our method generates topology-aware edge weights. Fig. 7 demonstrates a clear distribution shift between the edge weights connecting to ID and OOD nodes.

6.4 Ablation Study

In our framework, the reward consists of two terms: indicator function and entropy regularization term. To investigate the contribution of each term in the reward, we conduct an experiment in which only one term is available in the reward. The results are shown in Table 5 in Appendix. Indicator function aims to ensure the comparable accuracy achieved by the model. Without an indicator function, our method would experience a reduction in the accuracy on the ID nodes and the calibration error is also worsened. Without entropy regularization term in the reward, our method would be less effective to calibrate graph neural networks.

7 Conclusion

In this paper, we focus on calibrating graph neural networks (GNNs) on graphs containing OOD nodes. Noisy graphs exacerbate calibration errors of GNNs, and existing graph calibration methods become less effective. Inspired by graph structure learning, adjusting edge weights presents a plausible solution. However, assigning appropriate edge weights to a noisy graph is a nontrivial task. To address this challenge, we propose the Graph Calibration via Structure Optimization (GCSO) framework to derive an optimal, topology-aware graph structure. Extensive benchmark results demonstrate that our framework effectively calibrates GNNs in the presence of OOD nodes while maintaining comparable accuracy.

References

Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations (Poster)*, 2018.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pp. 2020–2027. PMLR, 2021.

Yujie Fang, Xin Li, Qianyu Chen, and Mingzhong Wang. Improving gnn calibration with discriminative ability: insights and strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11953–11960, 2024.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059. PMLR, 2016.

Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint*, 2019.

Arindam Ghosh, Thomas Schaaf, and Matthew Gormley. Adafocal: Calibration-aware adaptive focal loss. *Advances in Neural Information Processing Systems*, 35:1583–1595, 2022.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? *Advances in Neural Information Processing Systems*, 35:13775–13786, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. Policy-gnn: Aggregation optimization for graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 461–471, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

Wenqian Li, Yinchuan Li, Zhigang Li, Jianye Hao, and Yan Pang. DAG matters! gflownets enhanced explainer for graph neural networks. *CoRR*, abs/2303.02448, 2023.

Zenan Li, Qitian Wu, Fan Nie, and Junchi Yan. Graphde: A generative framework for debiased learning and out-of-distribution detection on graphs. *Advances in Neural Information Processing Systems*, 35: 30277–30290, 2022.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (Poster)*, 2016.

Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, pp. 1392–1403, 2022.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint*, 2013.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Caihua Shan, Yifei Shen, Yao Zhang, Xiang Li, and Dongsheng Li. Reinforcement learning enhanced explainer for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22523–22533, 2021.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint*, 2018.

Weili Shi, Xueying Yang, Xujiang Zhao, Haifeng Chen, Zhiqiang Tao, and Sheng Li. Calibrate graph neural networks under out-of-distribution nodes via deep q-learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 2270–2279, 2023.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395. Pmlr, 2014a.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 387–395. JMLR.org, 2014b.

Yu Song and Donglin Wang. Learning on graphs with out-of-distribution nodes. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1635–1645, 2022.

Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. *Advances in Neural Information Processing Systems*, 34:18033–18048, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15267–15275, 2024.

Linwei Tao, Minjing Dong, and Chang Xu. Dual focal loss for calibration. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 33833–33849. PMLR, 2023.

Leonardo Teixeira, Brian Jalaian, and Bruno Ribeiro. Are graph neural networks miscalibrated? *arXiv preprint*, 2019.

Yegor Tkachenko. Autonomous CRM control via CLV approximation with deep reinforcement learning in discrete and continuous action space. *CoRR*, abs/1504.01840, 2015.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Min Wang, Hao Yang, and Qing Cheng. Gcl: Graph calibration loss for trustworthy graph neural network. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 988–996, 2022.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 21681–21689, 2024.

Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, 34:23768–23779, 2021.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations (Poster)*, 2017.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, 2022.

Qitian Wu, Yiting Chen, Chenxiao Yang, and Junchi Yan. Energy-based out-of-distribution detection for graph neural networks. In *International Conference on Learning Representations*, 2023.

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 Web Conference 2024*, pp. 745–755, 2024a.

Hao Yang, Min Wang, Qi Wang, Mingrui Lao, and Yun Zhou. Balanced confidence calibration for graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3747–3757, 2024b.

Xueying Yang, Jiamian Wang, Xujiang Zhao, Sheng Li, and Zhiqiang Tao. Calibrate automated graph neural network via hyperparameter uncertainty. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4640–4644, 2022.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48. PMLR, 2016.

Jize Zhang, Bhavya Kailkhura, and T Yong-Jin Han. Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International Conference on Machine Learning*, pp. 11117–11128. PMLR, 2020.

Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836, 2020.

Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. Se-gsl: A general and effective graph structure learning framework through structural entropy optimization. In *Proceedings of the ACM Web Conference 2023*, pp. 499–510, 2023.

A Appendix

A.1 Algorithm

Algorithm 1 Algorithm of our GCSO framework

Input: input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, GNN backbone f, labels of the nodes Y, candidate nodes set $\mathcal{I}$, critic network $Q(s, a|\theta^Q)$, actor network $\pi(s|\theta^\pi)$, replay buffer $\mathcal{B}$, discount coefficient γ, hyperparameter α, initial noise σ_0, the total episode P, adjacent matrix $\mathbf{A}$.

Initialize the actor network π, critic network Q and replay buffer $\mathcal{B}$.

for 1,2,3..., P **do**

train the GNN backbone f with adjacent matrix A and obtain the $\text{acc}(B_m)$ and $\text{conf}(B_m)$ on validation nodes. Sample one target node u from the candidate nodes set $\mathcal{I}$.

obtain the edge set $\mathcal{E}^u = \{e_0^u, e_1^u, \cdots, e_{k-1}^u\}$ for each target node u.

for t from 1 to $|\mathcal{E}^u|$ **do**

calculate the discrepancy score for unsampled edges according to Eq. equation 4.

choose the edge e_t^u with the lowest discrepancy score and obtain the state s_t by Eq. equation 3.

calculate the adjusted edge weight $w_e = a_t$ from state s_t by Eq. equation 5.

add the noise to the adjusted edge weight for exploration via Eq. equation 8.

assign the adjusted edge weight to the original graph $\mathcal{G}$.

obtain the reward r from the GNN backbone f via Eq. equation 6.

form the transition tuple (s_t, a_t, s_{t+1}, r_t) into replay buffer $\mathcal{B}$.

randomly sample the data from replay buffer $\mathcal{B}$ and train the actor network π and critic network Q via Eq. equation 9 and Eq equation 10.

end for

generate the new edge weights and obtain new adjacent matrix A' using Eq. equation 5. Train the GNN backbone f and save the actor and critic networks based on the evaluation of model f.

update the adjacent matrix $\mathbf{A} = \mathbf{A}'$

end for

Table 5: Comparison between our method with complete reward signal, reward signal without entropy and reward signal without indicator function on Cora, PubMed, Coauthor-CS and Amazon-Computers in terms of node classification accuracy (Acc%)↑ and expected calibration error (ECE%)↓. The experiments are repeated 10 times and the average results and standard deviation are reported. Note that the primary focus of our study is the ECE performance of the methods.

Methods	Cora		PubMed		Coauthor-CS		Amazon-Computers	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
w/o entropy	84.88 ± 0.24	10.01 ± 0.60	**92.16** ± 0.61	1.55 ± 0.28	**92.80** ± 0.50	2.92 ± 0.46	90.91 ± 1.29	3.70 ± 0.54
w/o indicator	84.25 ± 0.27	10.08 ± 0.69	92.02 ± 0.16	1.57 ± 0.23	92.22 ± 0.35	3.11 ± 0.36	90.48 ± 0.75	3.61 ± 0.36
Complete	**84.95** ± 0.18	**9.90** ± 0.61	**92.16** ± 0.16	**1.49** ± 0.20	92.70 ± 0.47	**2.79** ± 0.33	**91.20** ± 0.48	**3.56** ± 0.49

A.2 Experiment

GCN (Kipf & Welling, 2016): The learning rate is 1e-2, weight decay is 5e-4. The hidden dimension is 64 with two layers. We choose Adam(Kingma & Ba, 2014) optimizer to train the model.

CaGCN (Wang et al., 2021) calibrates the confidence of the GNN by the post-hoc method to ensure the reliability of the prediction, with an estimation of different types of uncertainty. In the experiment we choose GCN (Kipf & Welling, 2016) as the base model. The hidden dimension is 64. The initial learning rate is 1e-2. The number of heads is 8.

GATS (Hsu et al., 2022): GATS proposed a new temperature scaling technique to calibrate the graph neural networks. We choose GCN (Kipf & Welling, 2016) as the base model. The hidden dimension is 64. In the experiment. The weight decay is 5e-3.

GCL (Wang et al., 2022): GCL loss function is proposed to calibrate the graph neural network in an end-to-end manner. The coefficient γ is set to 0.020. The hidden dimension is 64. The rest setting is the same as GCN (Kipf & Welling, 2016).

Table 6: The performance of GKDE-GCN on node classification and OOD node detection with old and new edge weights. The bold represents the best results.

Dataset	Edge weight	Acc(%)	ECE(%)	OOD AUROC(%)	OOD AUPR(%)
PubMed	original	**85.61**	10.73	85.21	**73.58**
	Modified	85.28	**9.70**	**85.39**	72.46
Citeseer	original	65.43	4.56	80.75	81.72
	Modified	**67.93**	**3.56**	**83.41**	**83.57**
Amazon Photo	original	89.83	3.05	69.05	61.35
	Modified	**91.42**	**1.80**	**69.90**	**62.16**

OODGAT (Song & Wang, 2022): OODGAT aims to perform the node classification and OOD detection simultaneously when the graph is mixed with OOD nodes. We adopt the same experimental setting as the original work. Note that the ID/OOD split in our experiment is different from that of OODGAT.

HyperU-GCN (Yang et al., 2022) focuses on automated graph learning which can obtain the optimal hyperparameters through joint optimization on model weights and hyperparameters. We adopt the same experimental setting as the original work.

GERDQ (Shi et al., 2023) investigates the calibration of graph neural networks when graph is mixed with OOD nodes. GERDQ aims to mitigate the calibration issue by adjusting the edge weight via deep Q-learning (Mnih et al., 2013).

DCGC (Yang et al., 2024a) is a data-centric method which aims to calibrate the graph neural networks by assigning larger weights to the decisive and homophilic edges.

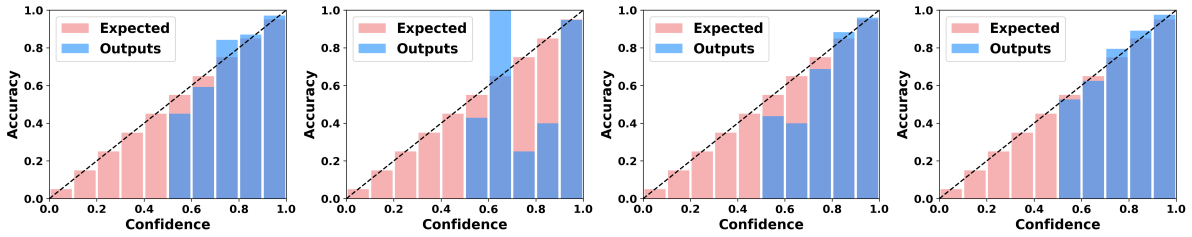
In our method, we adopt GCN (Kipf & Welling, 2016) and CaGCN (Wang et al., 2021) and GATS (Hsu et al., 2022) as our GNN backbone. The hyper-parameters of GCN are the same as the corresponding baselines. The learning rate is 1e-2 and weight decay is 5e-4. The hidden dimension is 64. The Actor and Critic in our framework are implemented as a three-layered MLP with the dimension of hidden layers 256 and 16, respectively. Adam (Kingma & Ba, 2014) is adopted for training optimization with the learning rate of 1e-3 for Actor and Critic, The weight decay is 1e-2. The α is set to 0.95, and the discount coefficient γ is 0.90. σ_0 is set to 0.2. The size of the replay buffer is 1e4 and the total number of episodes P is 30. The training epoch is 600 for all datasets. We choose 10 target nodes in the training and then select 16 edges for Cora and Citeseer (Yang et al., 2016), 128 edges for Arxiv (Hu et al., 2020) and 64 edges for the rest. We adopt 10 bins for evaluation of expected calibration error. All the experiments are running on the NVIDIA A5000. We test our method and baselines 10 times with different seeds and the average results are reported.

A.3 Case Study

We conduct a case study to investigate if the adjusted edge weights can improve the graph learning performance of other methods. GKDE-GCN (Zhao et al., 2020) is a representative method for detecting out-of-distribution (OOD) nodes by uncertainty estimation. We evaluate the performance of GKDE-GCN (Zhao et al., 2020) on node classification and OOD node detection with the adjusted edge weights learned by our framework. The metrics for the OOD node detection are AUROC and AUPR. We run the experiments 10 times on Cora, Citeseer, and PubMed (Yang et al., 2016), and report the average results in Table 6. The results show that our adjusted edge weights can help improve the node classification and OOD detection performance of the base model GKDE-GCN (Zhao et al., 2020).

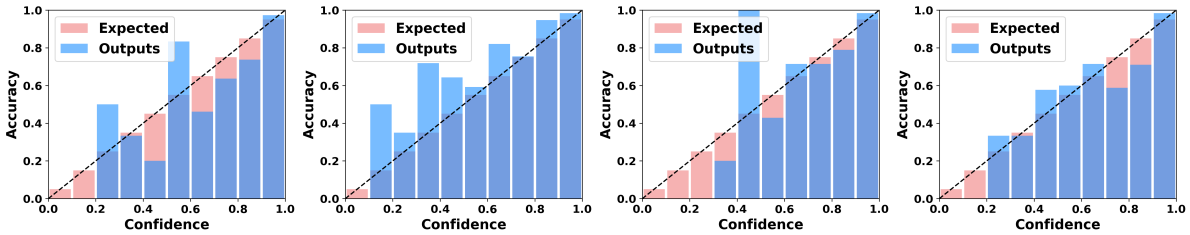
A.4 Visualization

To better visualize the ECE, the reliability diagrams of different methods on different datasets are illustrated from Fig. 8 to Fig. 11.



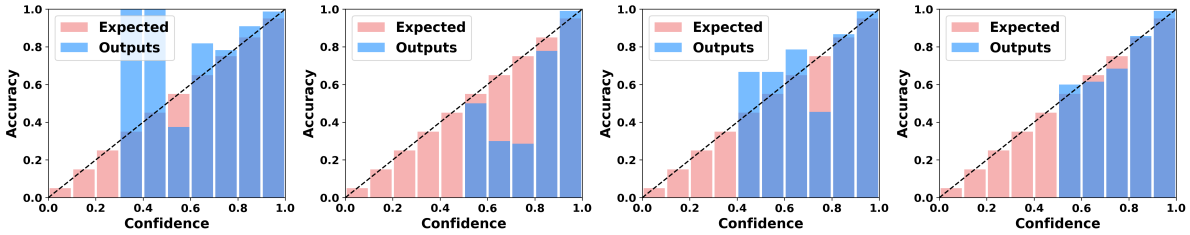
(a) GCN (b) HyperU-GCN (c) CaGCN (d) Ours

Figure 8: Reliability diagrams of different methods on PubMed with OOD nodes. Well-calibrated results would have closer alignment with the expected results along the diagonal line. The results suggest that the calibration issue is different and complicated on different datasets.



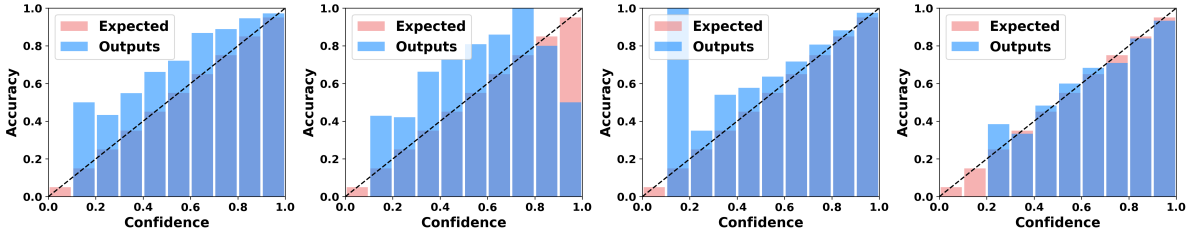
(a) GCN (b) HyperU-GCN (c) CaGCN (d) Ours

Figure 9: Reliability diagrams of different methods on Coauthor-CS with OOD nodes. Well-calibrated results would have closer alignment with the expected results along the diagonal line. The results suggest that the calibration issue is different and complicated on different datasets.



(a) GCN (b) HyperU-GCN (c) CaGCN (d) Ours

Figure 10: Reliability diagrams of different methods on Coauthor-Physics with OOD nodes. Well-calibrated results would have closer alignment with the expected results along the diagonal line. The results suggest that the calibration issue is different and complicated on different datasets.



(a) GCN (b) HyperU-GCN (c) CaGCN (d) Ours

Figure 11: Reliability diagrams of different methods on Arxiv with OOD nodes. Well-calibrated results would have closer alignment with the expected results along the diagonal line. The results suggest that the calibration issue is different and complicated on different datasets.