Calibrate Graph Neural Networks under Out-of-Distribution Nodes via Deep Q-learning

Weili Shi
rhs2rr@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Xueying Yang
xyang9@scu.edu
Santa Clara University
Santa Clara, California, USA

Xujiang Zhao
xuzhao@nec-labs.com
NEC Laboratories America
Princeton, New Jersey, USA

Haifeng Chen
Haifeng@nec-labs.com
NEC Laboratories America
Princeton, New Jersey, USA

Zhiqiang Tao
zhiqiang.tao@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Sheng Li
shengli@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

Graph neural networks (GNNs) have achieved great success in dealing with graph-structured data that are prevalent in the real world. The core of graph neural networks is the message passing mechanism that aims to generate the embeddings of nodes by aggregating the neighboring node information. However, recent work suggests that GNNs also suffer the trustworthiness issues. Our empirical study shows that the calibration error of the in-distribution (ID) nodes would be exacerbated if a graph is mixed with out-of-distribution (OOD) nodes, and we assume that the noisy information from OOD nodes is the root for the worsened calibration error. Both previous study and our empirical study suggest that adjusting the weights of edges could be a promising way to reduce the adverse impact from the OOD nodes. However, how to precisely select the desired edges and modify the corresponding weights is not trivial, since the distribution of OOD nodes is unknown to us. To tackle this problem, we propose a Graph Edge Re-weighting via Deep Q-learning (GERDQ) framework to calibrate the graph neural networks. Our framework aims to explore the potential influence of the change of the edge weights on target ID nodes by sampling and traversing the edges in the graph, and we formulate this process as a Markov Decision Process (MDP). Many existing GNNs could be seamlessly incorporated into our framework. Experimental results show that when wrapped with our method, the existing GNN models can yield lower calibration error under OOD nodes as well as comparable accuracy compared to the original ones and other strong baselines. The source code is available at: https://github.com/DamoSWL/Calibration-GNN-OOD.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Computing methodologies** → **Markov decision processes**.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614797

KEYWORDS

Graph neural networks; Reinforcement learning; Calibration of neural networks; Out-of-distribution generalization

ACM Reference Format:
Weili Shi, Xueying Yang, Xujiang Zhao, Haifeng Chen, Zhiqiang Tao, and Sheng Li. 2023. 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 (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614797

1 INTRODUCTION

Graph-structured data are ubiquitous in real world, such as social networks, infrastructure networks, and chemical molecules. Graph-structured data can not be easily handled by the traditional neural networks due to the non-Euclidean characteristics. To deal with graph-structured data, graph neural networks (GNN) [11, 18, 38] have been proposed, which aim to model the representative features of nodes by aggregating the information from neighbors through message passing. By effectively capturing semantic information from the graph, GNN can perform a wide range of tasks [10, 13, 14], such as node classification, link prediction, and graph classification. Up to now a series of graph neural networks have been proposed, such as Graph Convolutional Network (GCN) [18], GraphSAGE [11], and Graph Attention Network [38].

Similar to other types of neural networks, the reliability of GNN's predictions is an issue worthy of discussion. The previous study [9] revealed that complicated neural networks are prone to be over-confident, leading to an urgent need for calibrated predictions accounting for reliable and secure concerns. Focusing on graph neural networks, recent work [12, 36, 39] suggests that the prediction results from graph neural networks are also ill-calibrated, and the estimated probability yielded by GNN cannot represent true correctness likelihood of the data.

While the above work has conducted comprehensive investigation on calibrating graph neural networks, they always assume that graphs are "perfect" in the sense that node features are sampled from the same distribution. Yet, the calibration of GNNs would be getting more challenging when it comes to a more practical setting — learning on graphs with out-of-distribution nodes [33]. In the real world, graphs mixed with out-of-distribution (OOD) nodes are very common. For instance, in social networks, the ordinary people

are usually connected with some strangers such as salesperson or online scammers. Besides, a plenty of online fraudsters also exist in financial network and conduct illegal transaction with normal users. Unlike conventional graph learning, this new problem assumes that the graph contains not only the interested in-distribution (ID) nodes but also a large number of out-of-distribution (OOD) nodes. These OOD nodes do not fall into existing categories of the ID nodes, and their features are sampled from a different distribution. Previous work [33] suggests that the presence of OOD nodes has a negative impact on graph learning, since the noisy information from OOD nodes would be inevitably propagated to the ID nodes due to the message passing mechanism. With the presence of OOD nodes, our empirical experiment illustrates that the calibration error from the existing GNN models would be worsened.

To alleviate the negative impact from OOD nodes, one plausible method suggested by previous work [33] is to modify the weights of their edges connected to the known ID nodes. However, it is not trivial since the distribution of the OOD nodes in the graph is unknown to us. In this paper, we propose a Graph Edge Re-weighting via Deep Q-learning (GERDQ) framework to calibrate GNNs for graph learning with OOD nodes. Without any prior knowledge about the OOD nodes, our framework aims to explore the potential influence of the change of the edge weight on the target ID nodes by iteration of the sampled edges in the graph. To precisely select the appropriate edges and adjust the corresponding weights, we formulate the iteration of the edges as the Markov Decision Process (MDP) [1], in which we aim to obtain the optimal graph structure by utilizing the feedback from the GNN model with modified edge weights in the graph. In our framework, we adopt the deep Q-learning [29] to implement the MDP [1], and the feedback from the modified edge weights on the GNN can be modeled by the deep Q network. With our refined graph structure, the negative impact from the OOD nodes can be alleviated, and the calibration error of the GNN model can be reduced. Note that many existing GNN models can be incorporated into our framework. We evaluate our framework on six benchmark datasets and compare the performance of our framework with other baselines, some of which aim to calibrate the graph neural network by post-hoc calibration techniques or uncertainty-aware methods. The experimental results show that our method can provide reliable prediction results through the optimal graph structure and yield lower calibration errors than those of the original GNN models and other calibration-oriented methods. The major contributions of this work are summarized as follows:

- We propose a novel Graph Edge Re-weighting via Deep Q-learning (GERDQ) framework to solve the calibration issue of graph learning with OOD nodes. In the GERDQ framework, we explore the potential influence of the change of edge weight on the ID nodes through the edge iteration process. We formulate this process as Markov Decision Process (MDP) and implement it by deep Q-learning [29]. Our method can alleviate the negative impact of OOD nodes and achieve lower calibration errors under comparable accuracy with the adjusted edge weights.
- Existing GNN models can be seamlessly incorporated into our framework to obtain the optimal graph structure. Extensive experiments are conducted on six benchmark datasets under OOD settings. Results show that our proposed method could significantly improve the calibration performance with a comparable ID classification accuracy over the original GNN models and other calibration-oriented methods.

The rest of paper is arranged as follows. In Sec. 2 we introduce the related work on GNN calibration and reinforcement learning on graphs. In Sec. 3 we give the problem formulation and provide details of related concepts. In Sec. 4 we elaborate on our proposed framework and present the details of our algorithm. In Sec. 5 we show our experimental settings and compare our framework with baselines on benchmarks. At last, Sec. 6 concludes this paper.

2 RELATED WORK

2.1 Graph Structure Learning

Graph structure learning (GSL) aims to refine the graph structure for downstream tasks when the graph is incomplete or noisy [22, 23, 49]. With the optimized graph structure, the quality of graph representations can be restored. The core of GSL is the structure modeling that models the edge connectivity by an encoder. According to the structure modeling techniques [49, 50], the current work can be divided into metric-based approaches [2, 5, 40, 44], probabilistic-based approaches [6, 16, 28, 45, 47], and direct approaches [7, 15]. Besides, the latest work proposed SUBLIME [27] to obtain the optimal graph structure in a self-supervised manner. However, existing work does not consider the calibration issue of GNN when the graph is noisy. Our work aims to reduce the calibration error of GNN when the graph contains OOD nodes. We adopt the deep Q-learning [29] to refine the graph structure.

2.2 Graph Neural Network Calibration

The gap between the output probability and the ground-truth correctness likelihood of the predictions would result in the unreliability of GNN, which intrigues researchers to develop various methods to calibrate GNN. Recent work [39] suggests that the prediction of GNN is under-confident, and a calibration GNN model (CaGCN) [39] has been designed to learn the topology-aware post-hoc calibration function. The calibrated results would be obtained by the transformation imposed on the logits of GNN. HyperU-GCN [42] aims to model the hyperparameter uncertainty of graph neural networks with a bilevel formulation, and the prediction results of HyperU-GCN [42] could be calibrated by narrowing the uncertainty of hyperparameter. Most recently, researchers have identified several factors that influence the calibration of graph neural network through a comprehensive study on the calibration qualities of GNN [12]. And a novel calibration method named graph attention temperature scaling (GATS) [12] has been proposed accordingly. However, none of the aforementioned studies investigate the calibration issue of GNN under OOD settings. Our method improves the calibration of graph neural networks in the graph OOD problem setting.

2.3 Reinforcement Learning on Graphs

The rapid development of reinforcement learning (RL) has motivated researchers to develop RL-based methods for various graph

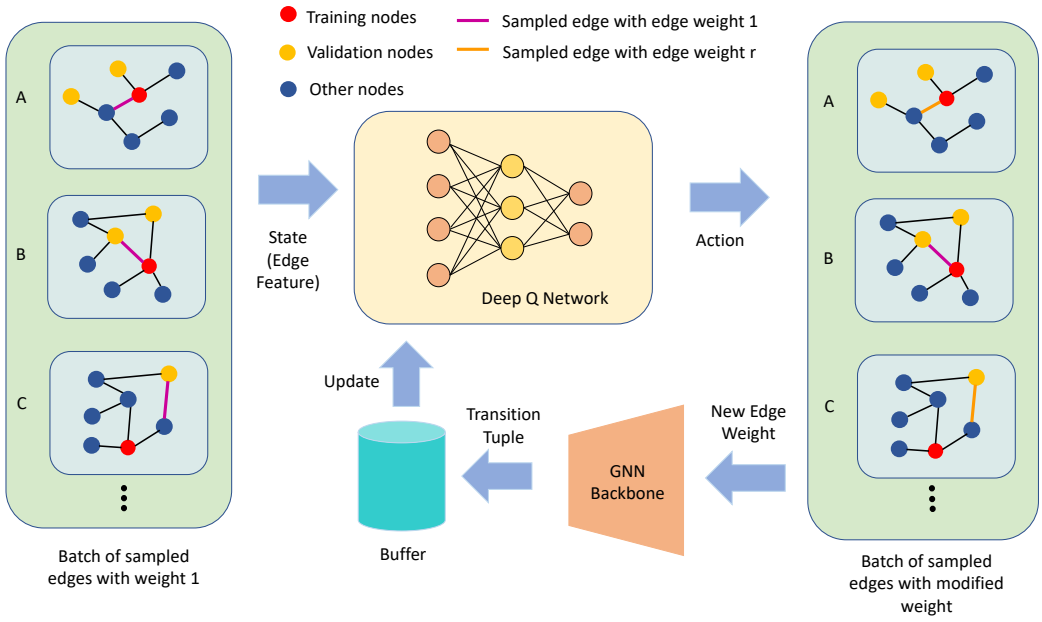


Figure 1: The illustration of our proposed Graph Edge Re-weighting via Deep Q-learning (GERDQ) framework. In our framework, we first sample the edges for iteration. We modify the weight of the sampled edges and investigate how the change of the edge weight affects the performance of ID node classification. We adopt the deep Q network to model the feedback from the change of the edge weight on the ID nodes, and the Q network is trained by deep Q-learning framework. After training, the Q network could produce the optimal graph structure. With the adjusted edge weights, the negative impact of OOD nodes can be alleviated, and the results can be well-calibrated. The other nodes denote the unlabelled ID nodes and OOD nodes.

learning tasks, such as neighborhood detection, information aggregation, and GNN explanation. Actor-Critic [19] algorithm and Deep Q-network (DQN) [30] are the most commonly used methods in reinforcement learning on graphs. For instance, Policy-GNN [20] aims to model the sampling procedure and message passing of GNNs under a meta-policy framework, which solves the challenge of determining the aggregation range of nodes in large-scale networks with the DQN algorithm. GraphNAS [8] explores all the possible configurations of graph neural network architectures with reinforcement learning, by maximizing the expected accuracy of the generated architectures on a validation dataset. Another line of work [24, 31] focuses on the explanation of graph neural networks through the subgraphs generated by reinforcement learning. In our work, we adopt the RL framework to obtain the optimal graph structure and alleviate the negative impact from the OOD nodes. With the modified edge weight, the results of graph neural networks can be calibrated implicitly.

2.4 Graph Learning with OOD Nodes

Up to now there are extensive work focusing on the graph learning with out-of-distribution nodes. These methods can be roughly categorized into out-of-distribution nodes detection [25, 26, 34, 46] and out-of-distribution generalization [3, 4, 21, 41]. Specifically, [33] proposed a new problem for graph learning with OOD nodes. In the new problem two major tasks are considered: semi-supervised outlier detection (SSOD) and semi-supervised node classification (SSNC). The tasks are challenging due to the unknown knowledge about the OOD nodes and the negative impact of OOD nodes on ID node classification. To tackle this problem, OODGAT [33] is proposed to perform both node classification and OOD detection. The OOD nodes can be identified by entropy estimation, and the performance on node classification can be improved by lowering the weight of edges between ID nodes and OOD nodes. However, OODGAT [33] does not consider the calibration issue of GNN under OOD setting. Our study aims to deal with the calibration problems of GNN modes with OOD nodes using reinforcement learning.

3 PRELIMINARY

3.1 Problem Formulation

Unlike the conventional close-world graph learning in which the nodes are sampled from the same distribution, similar to [33] we consider a realistic scenario in which the graph consists of both in-distribution (ID) nodes and out-of-distribution (OOD) nodes. A graph is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, in which $V = \{i | 1 \leq i \leq N\}$ denotes the node set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edge set. N is the total number of the nodes in the graph. The $\mathbf{X} \in \mathbb{R}^{N \times d}$ represents the feature matrix in which d is the dimension of node feature vector. The binary adjacency matrix is denoted by $A = \{0, 1\}^{N \times N}$. In our problem setting, the nodes set consists of both ID nodes and OOD nodes, i.e., $\mathcal{V} = \mathcal{V}_{ID} \cap \mathcal{V}_{OOD}$. The feature distribution of OOD nodes is different from 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 belong to any known category

of the ID nodes, and their labels are unknown to us. Similar to the conventional semi-supervised graph learning, the ID nodes can be divided into labelled ID nodes and unlabelled ID nodes $\mathcal{V}_{ID} = \mathcal{V}_{ID}^{l} \cap \mathcal{V}_{ID}^{ul}$. The goal of our problem is to obtain the a new graph structure $A' = (0,1]^{N\times N}$ and learn a classifier $f : X, A \rightarrow \tilde{Y}$ that not only achieves comparable performance on node classification but also yields well-calibrated prediction results with the presence of OOD nodes.

3.2 Expected Calibration Error

The expected calibration error (ECE) proposed by previous work [9] is a statistical measurement of the gap between the prediction probability and the ground-truth correctness likelihood from the output of the graph neural networks. In practice, 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. The expected calibrated error (ECE) can be defined as:

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

where

$$\begin{aligned} \text{acc}(B_m) &= \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\tilde{y}_i = y_i), \\ \text{conf}(B_m) &= \frac{1}{|B_m|} \sum_{i \in B_m} \tilde{p}_i. \end{aligned} \tag{2}$$

We adopt the expected calibration error as our major metric in the experiments.

3.3 Markov Decision Process

Markov Decision Process (MDP) [1] is a probability framework which is used to model a decision-making process, involving the interaction of an agent and an environment. A typical MDP can be formulated as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P_\pi, r, \gamma\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P_\pi(s'|s,a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state-action transition probability, r is the reward function, and $\gamma \in (0,1)$ is the discount factor. In MDP, the agent takes action a_t according to the current state s_t of the environment. The agent receives the reward r_t from the environment, and the current state s_t would be transited to next state s_{t+1} based on the transition probability. The goal of MDP is to learn the policy $\pi(a|s)$ that can maximize the discounted cumulative reward $J_\pi = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. To tackle the MDP problem, deep Q learning [29, 37] models the Q function derived from Bellman equation [35] using deep neural networks,

$$Q(s,a) = \mathbb{E}_{s' \sim \mathcal{S}}[r + \gamma \max_{a'} Q(s', a')]. \tag{3}$$

And the policy is obtained by $a = \text{argmax}_a Q(s,a)$. The advantage of deep Q-learning [29] is that it can handle the state with high dimension. The experience replay is introduced to remove data correlation. Besides, another work [37] introduced the double Q network (DQN) to ensure the stability and fast convergence of the training.

4 METHODOLOGY

Figure 1 illustrates the proposed GERDQ framework. Our framework consists of two major components: GNN backbone and deep Q network. In our framework, we first formulate the edge iteration process as a Markov Decision Process (MDP) and then evaluate the potential influence of the change of edge weight on the target ID nodes by the feedback from the GNN backbone. Similar to deep Q-learning, we adopt a deep neural network to model the Q function. By training on the sampled edges, our framework could effectively modify the weights of the desired edges and yield the optimal graph structure to reduce the negative impact from the OOD nodes. Consequently, the calibration error of GNN outputs could be reduced.

This section is arranged as follows. We first introduce our edge iteration process and then present the definitions of state, action, reward associated with this process. At last, we elaborate on the details of algorithms used in our framework.

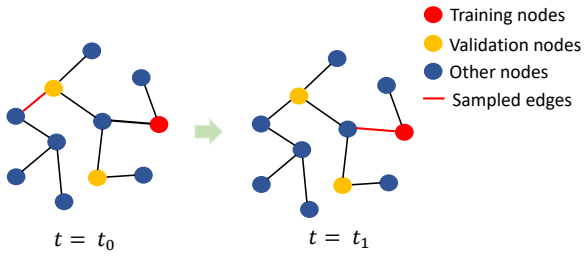


Figure 2: The illustration of iteration of the sampled edges from t_0 to t_1. We show a batch of one edge for brevity. The other nodes denote the unlabelled ID nodes and OOD nodes.

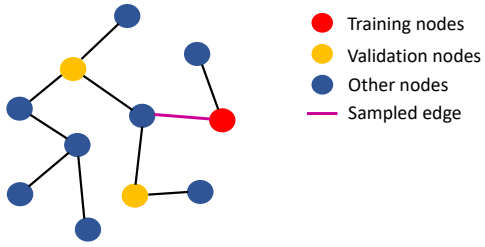


Figure 3: For the sampled edge, the average accuracy of the adjacent labelled nodes (denoted by red and yellow) would be adopted as the reward signal. The other nodes denote the unlabelled ID nodes and OOD nodes.

Algorithm 1 Algorithm for obtaining new edge set for training

Input: training nodes $\mathcal{V}_{train}$, validation nodes $\mathcal{V}_{val}$, hop value k, adjacent matrix A.
Output: k-hop neighbors edges $\mathcal{E}_k$ of training nodes and validation nodes.
$\mathcal{E}^k \leftarrow \{\}$.
for node v_i in $\mathcal{V}_{train} \cup \mathcal{V}_{val}$ **do**
  obtain k-hop neighbor edge $\mathcal{E}_i^k$ for node v_i using Algo. 2.
  $\mathcal{E}^k = \mathcal{E}^k \cup \mathcal{E}_i^k$.
end for

4.1 Edge Iteration Process

Algorithm 2 Algorithm for obtaining k-hop neighbor edges for a node

Input: adjacent matrix A, the node v_i, graph edge set $\mathcal{E}$.
Output: k-hop neighbor edges for the node v_i.
$\mathcal{E}^k \leftarrow \{\}$.
if $k \geq 0$ **then**
  obtain the 1-hop neighbor nodes $\mathcal{N}(v_i)$ for node v_i.
  for node $v_j \in \mathcal{N}(v_i)$ **do**
    $\mathcal{E}^k \leftarrow \mathcal{E}^k \cup \{e_{ij}\}$.
    obtain (k-1)-hop neighbor edges $\mathcal{E}^{k-1}$ for the node v_j.
    $\mathcal{E}^k \leftarrow \mathcal{E}^k \cup \mathcal{E}^{k-1}$.
  end for
end if

To sample the edges for training, we first choose the nodes with labels (i.e., training nodes and validation nodes), and form new edge set $\mathcal{E}^k$ by gathering the k-hop edges within each labelled node. The details are illustrated in Algorithm 1. At each timestep t during the training, we sample a batch of edges from the new edge set and modify their weights for evaluation of the influence of the adjusted weights on the target ID nodes, as shown in Figure 2. In order to enable our method smartly select the desired edges and change the corresponding weights, we formulate our edge iteration process as a Markov Decision Process, and we provide the definitions of the state, action and reward as follows.

State. We define our state $s \in \mathcal{S}$ as the edge features which is adopted as the average of the features of the connecting nodes.

Action. For each sampled edge, the action $a \in \{0, 1\}$ defined on the edge is the binary value that determines whether edge weight is modified or not. The action on the edge can be formulated as

$$a_{ij} = \begin{cases} 1 & \text{if } a = 1, \\ r & \text{if } a = 0, \end{cases} \tag{4}$$

where $r \in (0, 1)$ is assigned edge weight if action $a = 0$ is taken for the edge and a_{ij} denotes the weight of edge between node i and node j.

Reward. In our framework, the reward r is designed to reflect the influence of the change of edge weight on the target ID nodes and guide the training of the deep Q network to achieve the optimal graph structure. To ensure that the modified edge weights do not lower the performance of ID node classification, we formulate our reward signals as follows:

$$r = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\tilde{y}_i = y_i), \tag{5}$$

where N is the number of adjacent nodes of the sampled edge. $\tilde{y}_i$ and y_i denote the predicted label and ground-truth label of the node, respectively. As shown in Figure 3, since the change of weight of the sampled edge has equivalent impact on the adjacent nodes, the average accuracy of these in-distribution labelled nodes would be adopted as the reward signal. Our designed reward signal could guide the training of the deep Q network to obtain the optimal graph structure. With the refined edge weight, the GNN backbone can achieve the comparable performance for ID nodes, and the calibration error can be reduced due to the less noisy information propagated to ID nodes.

Algorithm 3 Algorithm for our proposed GERDQ framework

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, GNN backbone f, hop value $k = 2$, node features matrix $\mathbf{X}$, node labels $\mathbf{Y}$, training mask, validation mask, replay buffer B, epsilon probability ϵ, number of episode P, training steps T, Q network q.
form the new edge set for iteration and corresponding edges features with hop value k by Algo 1.
obtain the initial adjacent matrix A from graph $\mathcal{G}$.
for $p = 0, 1, 2, 3, ..., P$ **do**
  initialize and train GNN model f with the adjacent matrix A.
  for $t = 0, 1, 2, 3, ..., T$ **do**
    sample a batch of edges and form the state s_t.
    map the state into the action with the Q network q. Choose the action a_t for each edge according to the ϵ-greedy algorithm.
    update the adjacent matrix A with new edge weights by Eq. 4.
    calculate the reward r_t from the adjacent in-distribution labelled nodes via Eq. (5).
    sample next batch of edges as the next state s_{t+1}.
    store the transition tuple $D = (s_t, a_t, s_{t+1}, r_t)$ into replay buffer B.
    randomly sample the data from replay buffer B and train the Q network q by minimizing the loss function 6.
  end for
  update the weight of each edge by Q network and form the new adjacent matrix A'.
  update the adjacent matrix $A = A'$.
end for

4.2 Algorithm Details

The details of our framework are illustrated in Algorithm 3. We adopt the deep Q-learning [29, 37] method to train our Q network. The first step is to form the new edge set for iteration purpose. To ensure there are valid labelled ID nodes adjacent to the sampled edges and the reward signal is meaningful, we typically choose the edges 2-hop away from the labelled ID nodes. The state s is adopted as the edge features, and action a is chosen for the sampled edge according to the ϵ-greedy algorithm. The modified edge weight is given by Eq. (4). Then the reward r is obtained by Eq. (5). The transition tuple (s_t, a_t, s_{t+1}, r_t) at each timestep t is formed and stored in replay buffer. During the training, the batch of transition tuple is randomly sampled from the replay buffer to train the Q network via the loss function:

$$\mathcal{L}(\theta) = (r(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a))^2. \tag{6}$$

4.3 Discussions

In [12], the authors revealed five factors that may influence the calibration of the results from GNN model. One of the most important factor is the node similarity in the graph. [12] has demonstrated that higher node similarity yield lower calibration error and vice versa. In our problem setting, the presence of OOD nodes undermines the

Table 1: The statistics of datasets

Dataset	ID classes	OOD classes	#Nodes	#Edges	#Features	#Classes
Cora	[0 - 3]	[4 - 6]	2,708	10,556	1,433	7
Citeseer	[0 - 2]	[3 - 5]	3,327	9,104	3,703	6
PubMed	[0,1]	[2]	19,717	88,648	500	3
Amazon-Photo	[0 - 3]	[4 - 7]	7,650	238,162	745	8
Amazon-Computers	[0 - 4]	[5 - 9]	13,752	491,722	767	10
Coauthor-Physics	[0,1,2]	[3,4]	34,493	495,924	8,415	5

original node homophily and thus increases the calibration error for the GNN model. By lowering the weights of corresponding edges, we can prevent the noisy information propagated to the ID nodes and restore the original node similarity of the graph by making the OOD nodes less visible to the ID nodes. With the optimal graph structure obtained by our framework, the GNN model can achieve lower calibration errors and comparable accuracy for the graph with OOD nodes.

5 EXPERIMENTS

In this section, we mainly investigate the performance of our framework on the benchmark datasets and compare it with baselines. First, the experimental settings for graph learning with OOD nodes are introduced. We provide the performance of our proposed framework as well as the baseline methods. To visualize the calibration errors, the reliability diagrams of our framework and baselines are also provided. In the ablation study, we compare the results of our methods against that obtained by manually adjusting the edge weights given the prior knowledge of OOD node distribution.

5.1 Experimental Settings

Datasets. We evaluate our framework on six graph-structured datasets, namely Cora, Citeseer, PubMed [43], Amazon-Photo, Amazon-Computers, and Coauthor-Physics [32]. We basically adhere to the train/validation/test splits provided by previous work [32, 43]. Furthermore, to formulate scenario of the graph with OOD nodes, we manually split nodes of certain classes as out-of-distribution nodes, and the rest is regarded as in-distribution nodes. For instance, Cora [43] has 7 classes. Nodes from the first four classes are treated as in-distribution nodes while the others are OOD nodes. According to this split, the OOD nodes in the graph are masked out, and none of them would be used for training, validation, and testing. The details of the datasets are illustrated in Table 1.

Baselines. We compare our framework with six baselines, including GCN [18], HyperU-GCN [42], ST-GCN [48], CaGCN [39], GKDE-GCN [46], and OODGAT [33]. Among them, ST-GCN [48] focuses on automated graph learning which can obtain the optimal hyperparameters through joint optimization of model weights and hyperparameters. HyperU-GCN [42] can calibrate the prediction results implicitly by narrowing down the uncertainty of hyperparameters. CaGCN [39] 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, GKDE-GCN [46] is able to perform misclassification detection and OOD detection. OODGAT [33] is the first work that proposed graph learning with the OOD nodes. It performs both ID node classification and OOD node detection by identifying the entropy difference between ID nodes and OOD nodes. And the edge weight is adjusted accordingly to reduce the negative effect from OOD nodes. In our experiment, we only perform the ID node classification and compare the expected calibration error (ECE) under the comparable ID classification accuracy.

Metrics. The metrics we adopt in our experiments are accuracy and expected calibration error (ECE) [9]. ECE measures the gap between output probability and the ground-truth correctness likelihood of the data. A smaller value of ECE means the more reliable the prediction.

Implementation details. In our method, we adopt GCN [18] and HyperU-GCN [42] as our GNN backbone. For the training we follow the same setting of the GCN [18] and HyperU-GCN [42]. For HyperU-GCN, three dropout rates are in $[0, 0.9]$, one edge dropout rate in $[0, 0.9]$, and one weight decay in $[10^{-6}, 10^{-2}]$. The hypernetworks $(q_\phi(h),\ \theta(h))$ are implemented as a MLP with hidden dimension of 5 and 128, respectively. In our framework, the deep Q network is implemented as a two-layer MLP with hidden unit of 128. The training of Q network is optimized by Adam [17] algorithm with the learning rate of 5e-4. The episode steps P is set to 100. The hop values k is set to 2. The epsilon probability ϵ is initialized as 0.9, and the discount factor γ is set to 0.95. The updated edge weight r is set to 0.5. All the codes are implemented in Pytorch, and the experiment is running on NVIDIA RTX A5000. All the experiments are repeated 10 times, and the average values with standard deviations are reported.

5.2 Experimental Results

Table 2 and Table 3 summarize the experimental results of our framework and other baselines on the six benchmark datasets. The results show that baseline GNN models such as GCN [18] and ST-GCN [48] yield large calibration errors with the presence of OOD nodes, especially on Cora, Citeseer and PubMed [43]. Since these models treat the ID nodes and OOD nodes equally, the negative impact from the OOD nodes can not be reduced.

For the other baseline methods that aim to calibrate the prediction of graph neural network, their performance varies substantially on various datasets in our problem setting. HyperU-GCN [42] achieves worse calibration errors on Citeseer and PubMed [43] than those on other benchmarks. The performance of CaGCN-st [39] on the benchmark is also not satisfactory. For instance, it achieves 9.08% on Citeseer [43]. GKDE-GCN [46] can achieve the best ECE value on Citeseer, which is 3.74% on average. However, on the other datasets, it fails to provide a reliable prediction. For instance, on Cora [43], the average ECE GKDE-GCN is 9.88%, which is the

Table 2: Comparison between our proposed GERDQ framework and other baseline methods in terms of test accuracy (Acc%) and expected calibration error (ECE%) on Cora, Citeseer and PubMed datasets. The experiments are repeated 10 times, and the average results with standard deviations are reported. The bold represents the best results.

Methods	Cora		Citeseer		PubMed	
	Acc	ECE	Acc	ECE	Acc	ECE
GCN [18]	**86.87** ± 0.47	6.98 ± 0.25	71.20 ± 1.25	4.27 ± 0.63	92.28 ± 0.13	4.95 ± 0.25
ST-GCN [48]	81.71 ± 3.34	6.42 ± 1.30	62.80 ± 8.75	17.26 ± 4.86	90.99 ± 4.08	9.04 ± 2.91
HyperU-GCN [42]	85.19 ± 1.10	2.90 ± 4.09	70.86 ± 2.22	9.40 ± 3.47	92.98 ± 0.26	5.22 ± 1.43
CaGCN-st [39]	85.34 ± 0.43	4.54 ± 0.22	**72.52** ± 0.84	9.08 ± 4.87	92.29 ± 0.23	3.20 ± 0.33
GKDE-GCN [46]	85.38 ± 0.63	9.88 ± 0.39	67.09 ± 1.22	**3.74** ± 0.57	91.94 ± 0.26	4.29 ± 0.40
OODGAT [33]	75.46 ± 1.78	4.77 ±2.28	70.08 ± 2.98	12.45 ± 2.36	74.07 ± 1.19	9.35 ± 1.77
GERDQ+GCN (Ours)	85.01 ± 0.30	1.65 ± 0.19	70.90 ± 0.79	3.98 ± 0.54	**93.25** ± 0.48	**2.22** ± 0.54
GERDQ+HyperU (Ours)	85.22 ± 0.78	**1.28** ± 0.36	71.81± 1.53	3.94 ± 1.28	93.14 ± 0.56	2.59 ± 1.46

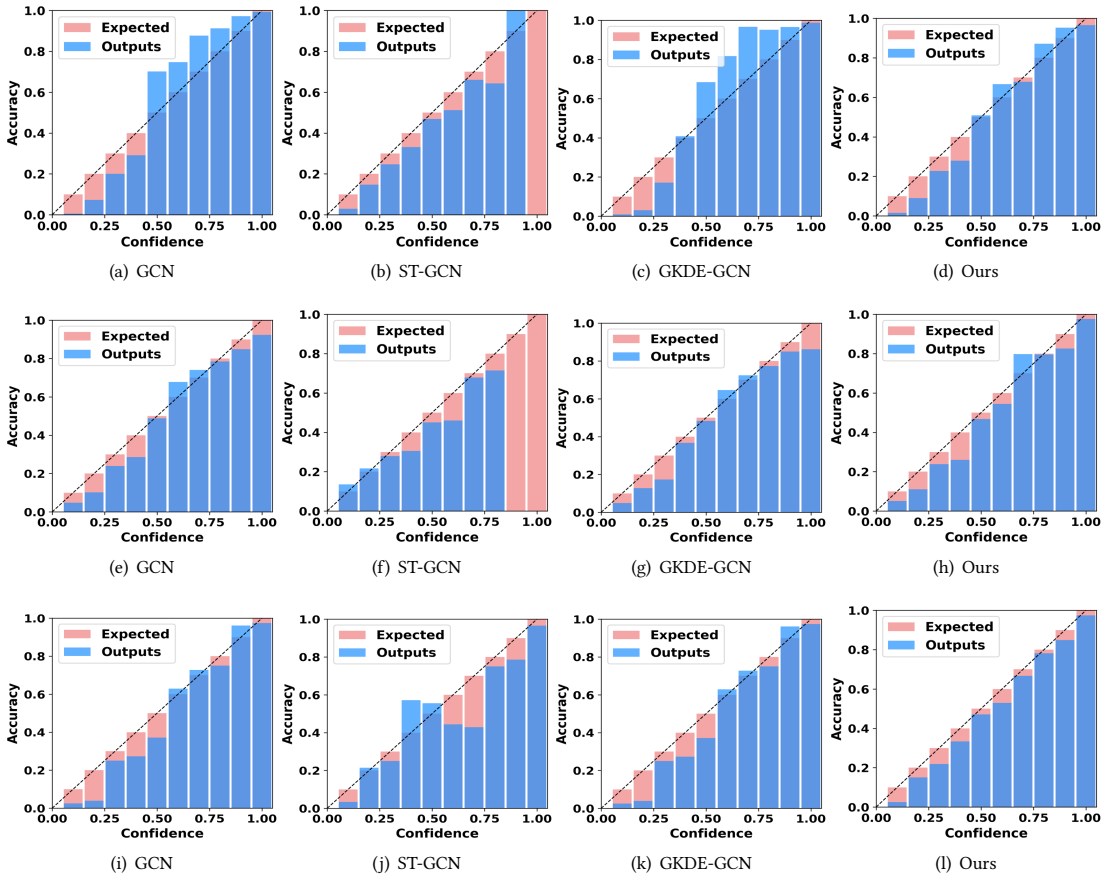
Table 3: Comparison between our proposed method and other baselines in terms of node classification accuracy (Acc%) and expected calibration error (ECE%) on Amazon-Photo, Amazon-Computers and Coauthor-Physics. The experiments are repeated 10 times, and the average results with standard deviations are reported. The bold represents the best results.

Methods	Amazon-Photo		Amazon-Computers		Coauthor-Physics	
	Acc	ECE	Acc	ECE	Acc	ECE
GCN [18]	91.11± 1.37	3.81 ± 0.64	83.80 ± 1.91	5.17 ± 1.13	93.94 ± 1.25	4.02 ± 0.94
ST-GCN [48]	92.93 ± 0.62	2.60 ± 0.32	89.66 ± 1.48	3.27 ± 0.26	96.92 ± 0.54	2.23 ± 0.40
HyperU-GCN [42]	91.56 ± 1.36	2.36 ± 0.89	89.37 ± 1.77	2.38 ± 0.56	97.06 ± 0.24	**1.25** ± 0.18
GKDE-GCN [46]	**93.35** ± 0.75	5.17 ± 0.65	88.29 ± 1.77	6.24 ± 0.89	**97.34** ± 0.29	4.73 ± 0.31
OODGAT [33]	91.66 ± 0.52	5.39 ±1.98	88.87 ± 0.40	**2.08** ± 0.71	95.28 ± 0.90	1.44 ± 0.38
GERDQ+GCN (Ours)	91.00 ± 0.80	3.39 ± 0.75	84.52 ± 1.76	4.68 ± 0.71	93.86 ± 1.22	3.66 ± 0.54
GERDQ+HyperU (Ours)	93.07 ± 0.71	**1.80** ± 0.50	**89.77** ± 0.88	**2.08** ± 0.44	96.83 ± 0.22	1.36 ± 0.12

Table 4: Comparison between our proposed method and original HyperU-GCN in terms of test accuracy (Acc%) and the expected calibration error (ECE%) on node classification. The numbers in the square bracket indicate the modified weight for the edge that is connecting the ID nodes and OOD nodes. The experiments are repeated 10 times, and the average results with standard deviations are reported. The bold represents the best results.

Methods	Cora		Citeseer		PubMed	
	Acc	ECE	Acc	ECE	Acc	ECE
HyperU-GCN[0.0]	85.84 ± 1.22	2.16 ± 2.01	**72.06** ± 0.96	10.14 ± 5.23	93.10 ± 0.41	8.02 ± 3.83
HyperU-GCN[0.3]	85.38 ± 0.73	4.20± 5.01	71.97 ± 1.18	9.36 ± 4.51	93.17 ± 0.53	5.30± 1.84
HyperU-GCN[0.5]	85.35 ± 0.79	3.48 ± 3.78	70.72 ± 2.21	8.47 ± 4.52	**93.33** ± 0.40	5.38 ± 1.05
HyperU-GCN[0.7]	85.10 ± 0.84	3.25 ± 4.06	71.72 ± 0.96	8.37 ± 4.98	93.17 ± 0.36	5.44 ± 0.99
HyperU-GCN[1.0]	85.19 ± 1.10	2.90 ± 4.09	70.86 ± 2.22	9.40 ± 3.47	92.98 ± 0.26	5.22 ± 1.43
HyperU-GCN[random]	**86.08** ± 0.82	1.41 ±0.45	71.72 ± 1.02	11.34 ± 4.30	92.86 ± 0.60	7.64 ± 6.08
GERDQ+HyperU (Ours)	85.22 ± 0.78	**1.28** ± 0.36	71.81 ± 1.53	**3.94** ± 1.28	93.14 ± 0.56	**2.59** ± 1.46

worst among all compared methods. And on larger benchmarks such as Amazon-Computers [32], the calibration error can also reach 6.24%. Note that our ID/OOD split is different from that in [33], and the performance of OODGAT [33] is quite inconsistent on the benchmarks. On Coauthor-Physics [32] it can achieve an calibration error of 1.44%. However, on other benchmarks such as PubMed [43] the calibration error reaches 9.35% on average. The experimental results suggest that neither the post-hoc calibration techniques nor the uncertainty-aware methods can consistently provide low calibration errors on all the benchmarks. The methods that aim to calibrate the result of GNN fail on some benchmarks when the graph is mixed with OOD nodes.



(a) GCN (b) ST-GCN (c) GKDE-GCN (d) Ours

(e) GCN (f) ST-GCN (g) GKDE-GCN (h) Ours

(i) GCN (j) ST-GCN (k) GKDE-GCN (l) Ours

Figure 4: Reliability diagrams of GCN, ST-GCN, GKDE-GCN and our method (GERDQ+HyperU) on (a-d) Cora, (e-h) Citeseer and (i-l) PubMed. Well-calibrated results would have closer alignment with the expected results along the diagonal line.

Our proposed framework calibrates the results of GNN by exploiting the feedback from the GNN with the change of the edge weights. The experiments show that when the GNN backbone is wrapped with our framework, it achieves better calibration error results on all datasets compared to the original one. For instance, on Cora [43], GERDQ+GCN can achieve an average calibration error of 1.65%, much lower than that achieved by the original GCN [18]. On some benchmarks, the ID node classification accuracy obtained by our method is also boosted compared to the original backbone. For instance, on Amazon-Photo [32] the accuracy of GERDQ+HyperU is approximately 1.5% higher than that of the original HyperU-GCN [42]. In most of the tasks, our GERDQ+HyperU can achieve the best calibration error with comparable ID node classification accuracy. The results validate the effectiveness of reinforcement learning strategy on adjusting graph edge weights to reduce the negative impact from OOD nodes.

To better illustrate the expected calibration errors, we draw the reliability diagram for GCN [18], ST-GCN [48], GKDE-GCN [46], and our proposed method (GERDQ+HyperU) on all the benchmarks. As shown in Figure 4 and Figure 5, the closer alignment of the output from the GNN model and expected results means a lower calibration error. We observe that for the baseline methods, some of the predictions are over-confident or under-confident and the outputs deviate from the diagonal line. The diagrams also show that our method achieves better alignment than other baselines.

5.3 Ablation Study

In the ablation study, we manually adjust the weight of edges that are connected the ID nodes and OOD nodes in HyperU-GCN [42] and compare the results with our proposed method, GERDQ+HyperU. The edge weights are manually set to 0, 0.3, 0.5, 0.7, and random numbers between 0 and 1. The results are illustrated in Table 4. By lowering the weight of edges that are connected the ID nodes and OOD nodes, the calibration error can be reduced on some datasets compared to settings when the edge weight is 1. However, their performance is still lower than that of our proposed method. Besides, the improvement is not consistent for the all chosen edge weights. The results suggest that simply lowering the edge weight on HyperU-GCN cannot necessarily ensure the reduction of calibration on all the datasets. The experiments suggest that the optimal graph structure obtained by our framework is more effective in reducing the calibration error than simply lowering the weight of targeted edges.

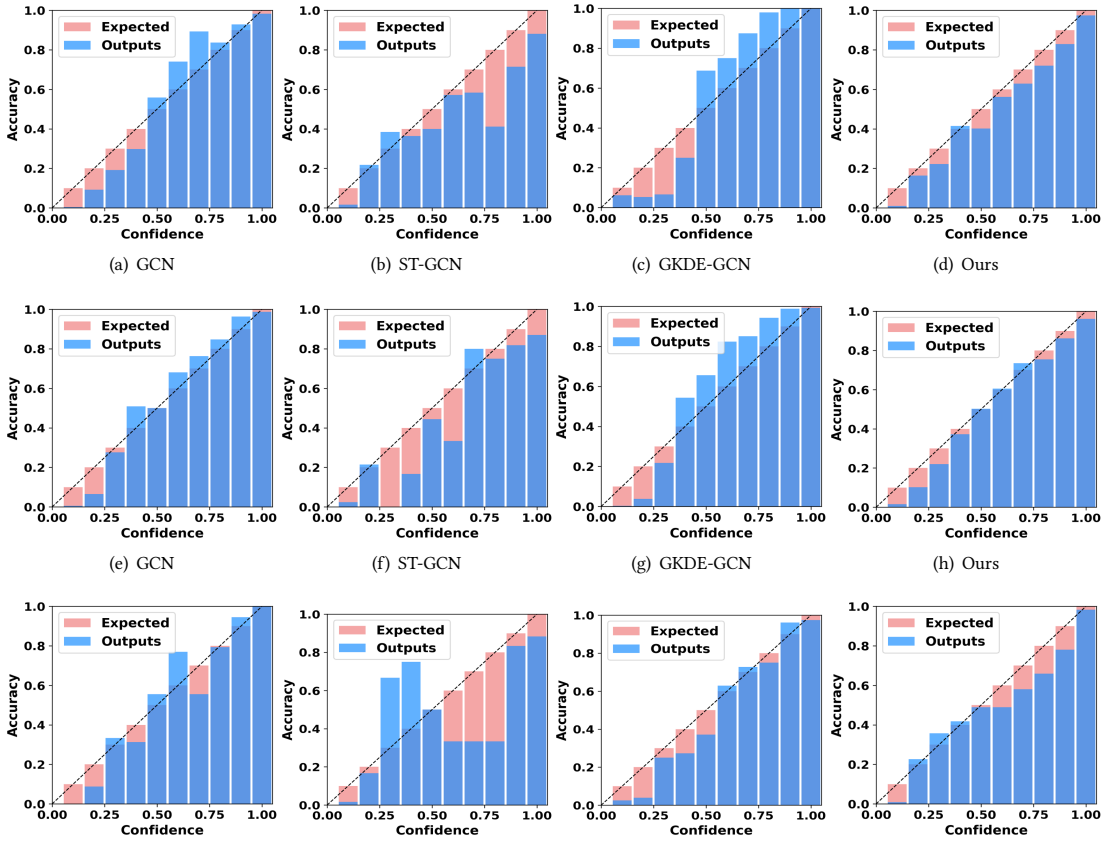


Figure 5: Reliability diagrams of GCN, ST-GCN, GKDE-GCN and our proposed method (GERDQ+HyperU) on (a-d) Amazon-Photo, (e-h) Amazon-Computers and (i-l) Coauthor-Physics. Well-calibrated results would have closer alignment with the expected results along the diagonal line.

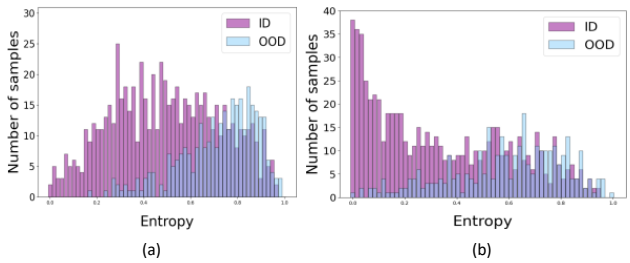


Figure 6: Distribution of ID nodes predictive uncertainties for (a) HyperU-GCN and (b) our proposed method (GERDQ+HyperU) on the Cora dataset.

Figure 6 depicts the distribution of nodes' predictive uncertainties of ID nodes and OOD nodes for original HyperU-GCN [42] and our proposed method (GERDQ+HyperU) on Cora [43]. The results show that when wrapped with our framework, the average entropy of the ID nodes achieved by our method is lower than that of original HyperU-GCN [42]. And the lower entropy indicates the lower uncertainty for the ID nodes,

6 CONCLUSION

In this paper, we have investigated the calibration of node classification on graphs with out-of-distribution (OOD) nodes. Existing methods that aim to calibrate the output of graph neural network on normal graph generally fail to provide reliable prediction in our problem setting due to the negative impact of OOD nodes. To tackle this challenge, we proposed a Graph Edge Re-weighting via Deep Q-learning framework. In our framework, we formulate the edge iteration as the Markov Decision Process and aim to achieve optimal graph structure by exploiting the feedback from the graph neural network with adjusted edge weights. With the refined edge weights, the negative impact from OOD nodes can be reduced and results from the GNN can be calibrated implicitly. Extensive experimental results on six benchmark datasets demonstrate that our method can achieve lower calibration errors with comparable classification accuracy compared to the baseline methods.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. W. Shi and S. Li are supported by the U.S. Army Research Office Award under Grant Number W911NF-21-1-0109.

REFERENCES

[1] Richard Bellman. 1957. Foundations of Markov Decision Processes. *Annals of Mathematics* 2, 2 (1957), 344–376.

[2] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems* 33 (2020), 19314–19326.

[3] Yongqiang Chen, Yonggang Zhang, Yatao Bian, Han Yang, MA Kaili, Binghui Xie, Tongliang Liu, Bo Han, and James Cheng. 2022. Learning causally invariant representations for out-of-distribution generalization on graphs. *Advances in Neural Information Processing Systems* 35 (2022), 22131–22148.

[4] Mucong Ding, Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Micah Goldblum, David Wipf, Furong Huang, and Tom Goldstein. 2021. A Closer Look at Distribution Shifts and Out-of-Distribution Generalization on Graphs. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

[5] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. SLAPS: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 22667–22681.

[6] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *Int'l Conf. on machine learning*. PMLR, 1972–1982.

[7] Xiang Gao, Wei Hu, and Zongming Guo. 2020. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 ieee Int'l Conf. on multimedia and expo (icme)*. IEEE, 1–6.

[8] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981* (2019).

[9] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Int'l Conf. on machine learning*. PMLR, 1321–1330.

[10] Dongliang Guo, Zhixuan Chu, and Sheng Li. 2023. Fair Attribute Completion on Graph with Missing Attributes. In *The Eleventh Int'l Conf. on Learning Representations*.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[12] Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. 2022. What Makes Graph Neural Networks Miscalibrated? *Advances in Neural Information Processing Systems* 35 (2022), 13775–13786.

[13] Xiaodong Jiang, Pengsheng Ji, and Sheng Li. 2019. CensNet: Convolution with Edge-Node Switching in Graph Neural Networks.. In *IJCAI*. 2656–2662.

[14] Xiaodong Jiang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. 2023. Co-Embedding of Nodes and Edges With Graph Neural Networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 45, 06 (2023), 7075–7086.

[15] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD Int'l Conf. on knowledge discovery & data mining*. 66–74.

[16] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M Bronstein. 2022. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 2 (2022), 1606–1617.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *Int'l Conf. on Learning Representations*.

[19] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

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

[21] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Learning invariant graph representations for out-of-distribution generalization. *Advances in Neural Information Processing Systems* 35 (2022), 11828–11841.

[22] Sheng Li and Yun Fu. 2014. Learning balanced and unbalanced graphs via low-rank coding. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2014), 1274–1287.

[23] Sheng Li, Hongfu Liu, Zhiqiang Tao, and Yun Fu. 2017. Multi-view graph learning with adaptive label propagation. In *IEEE Int'l Conf. on Big Data (Big Data)*. IEEE, 110–115.

[24] Wenqian Li, Yinchuan Li, Zhigang Li, Jianye Hao, and Yan Pang. 2023. DAG Matters! GFlowNets Enhanced Explainer For Graph Neural Networks. *CoRR* abs/2303.02448 (2023).

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

[26] Yixin Liu, Kaize Ding, Huan Liu, and Shirui Pan. 2023. Good-d: On unsupervised graph out-of-distribution detection. In *Proceedings of the Sixteenth ACM Int'l Conf. on Web Search and Data Mining*. 339–347.

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

[28] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM Int'l Conf. on web search and data mining*. 779–787.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

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

[32] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[33] Yu Song and Donglin Wang. 2022. Learning on Graphs with Out-of-Distribution Nodes. In *Proceedings of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*. 1635–1645.

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

[35] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[36] Leonardo Teixeira, Brian Jalaian, and Bruno Ribeiro. 2019. Are graph neural networks miscalibrated? *arXiv preprint arXiv:1905.02296* (2019).

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

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. [n. d.]. Graph Attention Networks. In *Int'l Conf. on Learning Representations*.

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

[40] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD Int'l Conf. on knowledge discovery & data mining*. 1243–1253.

[41] Yingxin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. 2021. Discovering Invariant Rationales for Graph Neural Networks. In *Int'l Conf. on Learning Representations*.

[42] Xueying Yang, Jiamian Wang, Xujiang Zhao, Sheng Li, and Zhiqiang Tao. 2022. Calibrate Automated Graph Neural Network via Hyperparameter Uncertainty. In *Proceedings of the 31st ACM Int'l Conf. on Information & Knowledge Management*. 4640–4644.

[43] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Int'l Conf. on machine learning*. PMLR, 40–48.

[44] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous graph structure learning for graph neural networks. In *Proceedings of the AAAI Conf. on artificial intelligence*, Vol. 35. 4697–4705.

[45] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *Proceedings of the aaai Conf. on artificial intelligence*, Vol. 35. 11015–11023.

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

[47] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *Int'l Conf. on Machine Learning*. PMLR, 11458–11468.

[48] Ronghang Zhu, Zhiqiang Tao, Yaliang Li, and Sheng Li. 2021. Automated graph learning via population based self-tuning GCN. In *Proceedings of the 44th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*. 2096–2100.

[49] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. 2021. A survey on graph structure learning: Progress and opportunities. *arXiv preprint arXiv:2103.03036* (2021).

[50] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036* 14 (2021).