ABSTRACT

In graph classification, attention- and pooling-based graph neural networks (GNNs) prevail to extract the critical features from the input graph and support the prediction. They mostly follow the paradigm of “learning to attend”, which maximizes the mutual information between the attended graph and the ground-truth label. However, this paradigm makes GNN classifiers recklessly absorb all the statistical correlations between input features and labels in the training data, without distinguishing the causal and noncausal effects of features. Instead of underscoring the causal features, the attended graphs are prone to visit the noncausal features as the shortcut to predictions. Such shortcut features might easily change outside the training distribution, thereby making the GNN classifiers suffer from poor generalization.

In this work, we take a causal look at the GNN modeling for graph classification. With our causal assumption, the shortcut feature serves as a confounder between the causal feature and prediction. It tricks the classifier to learn spurious correlations that facilitate the prediction in in-distribution (ID) test evaluation, while causing the performance drop in out-of-distribution (OOD) test data. To endow the classifier with better interpretation and generalization, we propose the Causal Attention Learning (CAL) strategy, which discovers the causal patterns and mitigates the confounding effect of shortcuts. Specifically, we employ attention modules to estimate the causal and shortcut features of the input graph. We then parameterize the backbone adjustment of causal theory — combine each causal feature with various shortcut features. It encourages the stable relationships between the causal estimation and the prediction, regardless of the changes in shortcut parts and distributions. Extensive experiments on synthetic and real-world datasets demonstrate the effectiveness of CAL. Codes are released at https://github.com/yongduousui/CAL.

CCS CONCEPTS

• Mathematics of computing → Graph algorithms; • Computing methodologies → Learning latent representations.

KEYWORDS

Graph Neural Networks, Graph Classification, Causal Intervention

1 INTRODUCTION

Graph neural networks (GNNs) [8, 17] have exhibited impressive performance of graph classification across various domains, such as chemical molecules, social networks, and transaction graphs. Such a success mainly comes from the powerful representation learning of GNNs, which incorporates the graph structure and encodes them into the representations in an end-to-end way. Hence, it is crucial to emphasize the critical part of the input graph, while filtering the trivial part out [23, 42, 45]. For example, when classifying the mutagenic property of a molecular graph [24], GNNs are expected to latch on the functional groups (i.e., nitrogen dioxide (NO2)), instead of the irrelevant patterns (i.e., carbon rings) [5, 44]; when detecting fraud in a transaction network, malicious behaviors or coalitions of users are more informative than benign features.

Towards specifying the critical parts in graphs, some follow-on studies [10, 16, 35, 43] adopt the paradigm of “learning to attend” [34, 38] — maximizing the mutual information between the attended graph and the ground-truth label — to find the attended graph that maximizes the predictive performance. Specifically, there are two research lines in this paradigm: (1) Attention-based methods [4, 16, 22, 33, 35]. They often utilize the attention modules for nodes or edges to locate the attended graphs. These attention modules act like soft masks to identify the importance of each edge and node to the final representations and predictions. (2) Pooling-based methods [10, 19, 43, 48]. They directly adopt hard masks to select a subset of nodes or edges as the attended graphs, to perform the information propagations. These attended graphs aim to approach
the features that are beneficial for minimizing the training loss, instead of distinguishing the causal and noncausal effects. Unfortunately, recent efforts [2, 11, 12, 18] have shown that the current attention or pooling learning methods are prone to exploit the shortcut features to make decisions. These shortcuts usually come from the data selection biases, noisy features, or some trivial patterns from graphs, which are noncausal but discriminative in training data. Due to the existence of these shortcuts, models can capture shortcut features to finish the classification tasks without struggling to learn causal features. For example, instead of probing into the causal effect of the functional groups, the attended graphs prefer “carbon rings” as the cues of the “mutagenic” class, because most training “mutagenic” molecules are in the “carbon rings” context. While such correlations represent statistical relations inherent in the training data and are beneficial to the in-distribution (ID) test evaluations, they inevitably cause a huge performance drop in the out-of-distribution (OOD) test data that are at odds with the training distribution. Taking the molecule classification as an example again, when most test “non-mutagenic” molecules appear in the “carbon rings” context, the attended graphs mislead the GNNs to predict “mutagenic” while reducing the confounding effect of the attended graph. We then pool the shortcut patterns to make predictions. With our causal assumption in Figure 1, the shortcut feature as a confounder. Then we propose the causal attention mechanism to select the most attentive node representations:

\[
\mathbf{H}' = \text{GConv} (\mathbf{A} \circ \mathbf{M}_u, \mathbf{H})
\]

where \(\mathbf{M}_u \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|} \) denotes the attention matrix that is often derived from trainable parameters and node representations. For node-level attention, several studies [18, 19, 22] define the self-attention mask to select the most attentive node representations:

\[
\mathbf{H}' = \text{GConv} (\mathbf{A}, \mathbf{H} \circ \mathbf{M}_u)
\]

where \(\mathbf{M}_u \in \mathbb{R}^{|\mathcal{V}| \times 1} \) represents the node-level attentions, which can be generated by a network (e.g., GNNs or MLPs); \(\circ\) is the broadcasted element-wise product. Hereafter, we can make further pooling operation [19] for the output node representations \(\mathbf{H}_{\text{out}}\) and summarize the graph representation \(\mathbf{h}_G\) for graph \(G\) via the readout function \(f_{\text{readout}}(\cdot)\). Then we use a classifier \(\Phi\) to project the graph representation into a probability distribution \(\mathbf{z}_G\):

\[
\mathbf{h}_G = f_{\text{readout}}\left(\{\mathbf{h}_{i,\text{out}}^u | i \in \mathcal{V}\}\right), \quad \mathbf{z}_G = \Phi(\mathbf{h}_G).
\]

These methods follow the paradigm of “learning to attend” by minimizing the following empirical risk:

\[
\mathcal{L}_\text{CE} = -\frac{1}{|\mathcal{D}|} \sum_{G \in \mathcal{D}} y_G \log(\mathbf{z}_G).
\]

where \(\mathcal{L}_\text{CE}\) is the cross-entropy loss over the training data \(\mathcal{D}\), and \(y_G\) is the ground-truth label vector of \(G\). However, this learning strategy heavily relies on the statistical correlations between the input graphs and labels. Hence, they will inevitably capture the noncausal shortcut features to make predictions.

3 METHODOLOGY

In this section, we first analyze the GNN learning from the perspective of causality. From our causal assumption, we identify the shortcut feature as a confounder. Then we propose the causal attention learning strategy to alleviate the confounding effect.
We have realized that shielding the GNNs from the confounder is the key to exploiting causal features. Instead of modeling the confounded \( P(Y|C) \) in Figure 1, we should achieve the graph representation learning by eliminating the backdoor path. But how to achieve this? Fortunately, causal theory [26, 27] provides us with a feasible solution: we can exploit the do-calculus on the variable \( C \) to remove the backdoor path by estimating \( P_m(Y|C) = P(Y|do(C)) \). It needs to stratify the confounder \( S \) between \( C \) and \( Y \). Therefore, we can obtain the following three essential conclusions:

- The marginal probability \( P(S = s) \) is invariant under the intervention, because the shortcut feature will not be affected by cutting off the backdoor path. Thus, \( P(s) = P_m(s) \).
- The conditional probability \( P(Y|C, s) \) is invariant, because \( Y \)'s response to \( C \) and \( S \) has nothing to do with the causal effect between \( C \) and \( S \). Then we can get: \( P_m(Y|C, s) = P(Y|C, s) \).
- Obviously, the variables \( C \) and \( S \) are independent under the causal intervention, which we have: \( P_m(s|C) = P_m(s) \).

Based on the above conclusions, we have:

\[
\begin{align*}
P(Y|do(C)) &= P_m(Y|C) \\
&= \sum_{s \in T} P_m(Y|C, s)P_m(s|C) \quad \text{(Bayes Rule)} \\
&= \sum_{s \in T} P_m(Y|C, s)P_m(s) \quad \text{(Independency)} \\
&= \sum_{s \in T} P(Y|C, s)P(s),
\end{align*}
\]

where \( T \) denotes the confounder set; \( P(Y|C, s) \) represents the conditional probability given the causal feature \( C \) and confounder \( s \); \( P(s) \) is the prior probability of the confounder. Equation (5) is usually called backdoor adjustment [26], which is a powerful tool to eliminate the confounding effect. However, there exist two challenges for implementing Equation (5): i) The confounder set \( T \) is commonly unobservable and hard to obtain. ii) Due to the discrete nature of graph data, it seems difficult to directly manipulate the graph data, conditioning on domain-specific constraints (e.g., valency rules in molecule graphs). In section 3.4.3, we will introduce a simple yet effective solution to overcome these issues.

### 3.2 Backdoor Adjustment

We have realized that shielding the GNNs from the confounder \( S \) is the key to exploiting causal features. Instead of modeling the confounded \( P(Y|C) \) in Figure 1, we should achieve the graph representation learning by eliminating the backdoor path. But how to achieve this? Fortunately, causal theory [26, 27] provides us with a feasible solution: we can exploit the do-calculus on the variable \( C \) to remove the backdoor path by estimating \( P_m(Y|C) = P(Y|do(C)) \). It needs to stratify the confounder \( S \) between \( C \) and \( Y \). Therefore, we can obtain the following three essential conclusions:

- The marginal probability \( P(S = s) \) is invariant under the intervention, because the shortcut feature will not be affected by cutting off the backdoor path. Thus, \( P(s) = P_m(s) \).
- The conditional probability \( P(Y|C, s) \) is invariant, because \( Y \)'s response to \( C \) and \( S \) has nothing to do with the causal effect between \( C \) and \( S \). Then we can get: \( P_m(Y|C, s) = P(Y|C, s) \).
- Obviously, the variables \( C \) and \( S \) are independent under the causal intervention, which we have: \( P_m(s|C) = P_m(s) \).

Based on the above conclusions, we have:

\[
\begin{align*}
P(Y|do(C)) &= P_m(Y|C) \\
&= \sum_{s \in T} P_m(Y|C, s)P_m(s|C) \quad \text{(Bayes Rule)} \\
&= \sum_{s \in T} P_m(Y|C, s)P_m(s) \quad \text{(Independency)} \\
&= \sum_{s \in T} P(Y|C, s)P(s),
\end{align*}
\]

where \( T \) denotes the confounder set; \( P(Y|C, s) \) represents the conditional probability given the causal feature \( C \) and confounder \( s \); \( P(s) \) is the prior probability of the confounder. Equation (5) is usually called backdoor adjustment [26], which is a powerful tool to eliminate the confounding effect. However, there exist two challenges for implementing Equation (5): i) The confounder set \( T \) is commonly unobservable and hard to obtain. ii) Due to the discrete nature of graph data, it seems difficult to directly manipulate the graph data, conditioning on domain-specific constraints (e.g., valency rules in molecule graphs). In section 3.4.3, we will introduce a simple yet effective solution to overcome these issues.

### 3.3 Causal and Trivial Attended-graph

Given a graph \( G = \{X, A\} \), we formulate the soft masks on the graph structure and node feature as \( M_a \in \mathbb{R}^{|V| \times |V|} \) and \( M_x \in \mathbb{R}^{|V| \times 1} \), respectively. Wherein, each element of the masks indicates the attention score relevant to the task of interest, which often falls into the range of \((0, 1)\). Given an arbitrary mask \( M \), we define its complementary mask as \( \overline{M} = 1 - M \), where \( 1 \) is the all-one matrix.

Therefore, we can divide the full graph \( G \) into two attended-graphs: \( G_1 = \{A \odot M_a, X \odot M_x\} \) and \( G_2 = \{A \odot \overline{M}_a, X \odot \overline{M}_x\} \).

With the inspection on the data-generating process, recent studies [18, 23, 42, 45] argue that the label of a graph is usually determined by its causal part. Considering a molecular graph, its mutagenic property relies on the existence of relevant functional groups [45]. Taking the digit image in the form of a superpixel graph as another example, the coalition of digit-relevant nodes determines its label. Formally, given a graph \( G \), we define the attended graph collecting all causal features as the causal attended-graph \( G_c \), while the counterpart forms the trivial attended-graph \( G_t \). However, the ground-truth attended-graph is usually unavailable in real-world applications. Hence, we aim to capture the causal and trivial attended-graph from the full graph by learning the masks: \( G_c = \{A \odot M_a, X \odot M_x\} \) and \( G_t = \{A \odot \overline{M}_a, X \odot \overline{M}_x\} \). Learning to identify causal attended-graphs not only guides the representation learning of GNNs, but also answers “What knowledge does the GNN use to make predictions?”, which is crucial to the applications on explainability, privacy, and fairness.

### 3.4 Causal Attention Learning

To implement the aforementioned backdoor adjustment, we propose the Causal Attention Learning (CAL) framework:

#### 3.4.1 Estimating soft masks.

Towards effective causal intervention, it is necessary to separate the causal and shortcut features from the full graphs. To this end, we hire attention modules, which yield two branches for the causal and trivial proposals. Given a GNN-based encoder \( f(\cdot) \) and a graph \( G = \{A, X\} \), we can obtain
the node representations:
\[ H = f(A, X). \]  
(6)

Then we adopt two MLPs: \( \text{MLP}_{\text{node}}(\cdot) \) and \( \text{MLP}_{\text{edge}}(\cdot) \) to estimate the attention scores from two orthogonal perspectives: node-level and edge-level. For node \( v_i \) and edge \( (e_i, v_j) \) we can obtain:
\[ \alpha_{c_i}, \alpha_{t_i} = \sigma(\text{MLP}_{\text{node}}(h_i)), \]  
(7)
\[ \beta_{c_{ij}}, \beta_{t_{ij}} = \sigma(\text{MLP}_{\text{edge}}(h_i|h_j)). \]  
(8)
where \( \sigma(\cdot) \) is softmax function, \( \| \) denotes concatenation operation; \( \alpha_{c_i}, \alpha_{t_i} \) represent the node-level attention score for node \( v_i \) and edge-level attention score for edge \( (v_i, v_j) \) in causal attended-graph; analogously, \( \beta_{c_{ij}}, \beta_{t_{ij}} \) are for trivial attended-graph. Note that \( \alpha_{c_i} + \alpha_{t_i} = 1 \) and \( \beta_{c_{ij}} + \beta_{t_{ij}} = 1 \). These attention scores indicate how much the model pays attention to each node or edge in the corresponding attended-graph. Now we can construct the soft masks \( \mathbf{M}_c \), \( \mathbf{M}_t \), \( \mathbf{M} = \mathbf{M}_c \cup \mathbf{M}_t \) based on the attention scores \( \alpha_{c_i}, \alpha_{t_i}, \beta_{c_{ij}}, \beta_{t_{ij}} \), respectively. Finally, we can decompose the original graph \( G \) into the initial causal and trivial attended-graphs: \( \hat{G}_c = \{A \odot \mathbf{M}_c, X, M_X\} \) and \( \hat{G}_t = \{A \odot \mathbf{M}_t, X \odot \mathbf{M}_t\} \).

### 3.4.2 Disentanglement

Until now, we have distributed the attention scores at the granularity of nodes and edges to create the initial attended-graphs. Now we need to make the causal and trivial attended-graphs to capture the causal and shortcut features from the input graphs, respectively. Specifically, we adopt two GNN layers to obtain the representations of attended-graphs and make predictions via readout function and classifiers:
\[ h_{\hat{G}_c} = f_{\text{readout}}(\text{GraphConv}(A \odot \mathbf{M}_c, X, M_X)), \quad \hat{z}_{\hat{G}_c} = \Phi_c(h_{\hat{G}_c}). \]  
(9)
\[ h_{\hat{G}_t} = f_{\text{readout}}(\text{GraphConv}(A \odot \mathbf{M}_t, X \odot \mathbf{M}_t)), \quad \hat{z}_{\hat{G}_t} = \Phi_t(h_{\hat{G}_t}). \]  
(10)

The causal attended-graph aims to estimate the causal features, so we classify its representation to the ground-truth label. Thus, we define the supervised classification loss as:
\[ L_{\text{sup}} = -\frac{1}{|D|} \sum_{G \in D} \mathbb{I}(y_G = \hat{z}_G). \]  
(11)

where \( L_{\text{sup}} \) is the cross-entropy loss over the training data \( D \). The trivial attended-graph aims to approach the trivial patterns that are unnecessary for classification. Hence, we push its prediction evenly to all categories and define the uniform classification loss as:
\[ L_{\text{unif}} = \frac{1}{|D|} \sum_{G \in D} \text{KL}(y_G, \hat{z}_G), \]  
(12)
where \( \text{KL} \) denotes the KL-Divergence, \( y_G \) represents the uniform distribution. By optimizing the above two objectives, we can effectively disentangle causal and trivial features. Please note that prior efforts [18, 23, 42, 45] have shown that the mutual information between the causal part and label is greater than that between the full graph and label, due to the widespread trivial patterns or noise. Hence, the proposed disentanglement will not make the captured causal attended-graph converge to the full graph (noiseless full graph is a special case), which is not an optimal solution. See Section 4.5 for more supporting evidence and analyses.

### 3.4.3 Causal intervention

As shown in Equation (5), one promising solution to alleviating the confounding effect is the backdoor adjustment — that is, stratifying the confounder and pairing the target causal attended-graph with every stratification of trivial attended-graph to compose the “intervened graphs”. However, due to the irregular graph data, it is impossible to make the intervention on data-level, e.g., changing a graph’s trivial part to generate a counterfactual graph data. Towards this end, we make the implicit intervention on representation-level and propose the following loss guided by the backdoor adjustment:
\[ z_{G'} = \Phi(h_{\hat{G}_c} + h_{\hat{G}_t}), \]  
(13)
\[ L_{\text{caus}} = -\frac{1}{|D|} \sum_{G \in D} \sum_{G' \in \hat{T}} \mathbb{I}(y_G' = \hat{z}_G), \]  
(14)
where \( z_{G'} \) is the prediction from a classifier \( \Phi \) on “implicit intervened graph” \( G' \); \( h_{\hat{G}_c} \) is the representation of causal attended-graph \( \hat{G}_c \) derived from Equation (9); while \( h_{\hat{G}_t} \) is the representation of stratification \( \hat{G}_t \) obtained via Equation (10); \( \hat{T} \) is the estimated stratification set of the trivial attended-graph, which collects the appearing trivial features from training data. In practice, we apply random addition to make the intervention in Equation (14). We define the Equation (14) as the causal intervention loss. It pushes the predictions of such intervened graphs to be invariant and stable across different stratifications, due to the shared causal features. Finally, the objective of CAL can be defined as the sum of the losses:
\[ L = L_{\text{sup}} + \lambda_1 L_{\text{unif}} + \lambda_2 L_{\text{caus}} \]  
(15)
where \( \lambda_1 \) and \( \lambda_2 \) are hyper-parameters that determine the strength of disentanglement and causal intervention, respectively. The detailed algorithm of CAL is provided in Appendix A.1, Alg.1, and the overview of CAL is depicted in Figure 2.

### 4 EXPERIMENTS

To verify the superiority and effectiveness of the proposed CAL, we conduct experiments to answer the following research questions:

- **RQ1**: How effective is the proposed CAL in alleviating the out-of-distribution (OOD) issue?
4.1 Experimental Settings

4.1.1 Datasets. We conduct experiments on both synthetic datasets and real-world datasets.

- **Synthetic graphs**: Following [42], we create the synthetic dataset for graph classification, which contains a total of 8,000 samples with 4 classes, and keeps balance (2,000 samples) for each class. As shown in Figure 3, each sample consists of two parts: causal subgraph and trivial subgraph. More details about the causal and trivial subgraph are provided in Appendix A.2. The task is to predict the type of the causal part in the whole graph. For simplicity, we choose the "House" class to define the bias-level:

\[
    b = \frac{\text{#Tree-House}}{\text{#House}} \tag{16}
\]

where #Tree-House denotes the number of "House" causal subgraphs with the "Tree" trivial subgraphs, and #House presents the number of graphs in the "House" class, which is 2,000. We set the proportion of "Tree" in the other three classes to 1 − b. Obviously, for the unbiased dataset, b = 0.5. We abbreviate the synthetic dataset with bias-level b as SYN-b. We keep the same bias-level on the training/validation set and keep the testing set unbiased. Please refer to Appendix A.2 for more details.

- **Real-world graphs**: We conduct experiments on three biological datasets (MUTAG, NCI1, PROTEINS), three social datasets (COLLAB, IMDB-B, IMDB-M) [24], and two superpixel datasets (MNIST, CIFAR-10) [18]. More details, such as statistics and splitting of datasets, are provided in Appendix A.2.

4.1.2 Baselines. To verify the superiority of CAL, we adopt the following prevalent graph classification solutions as baselines:

- **Attention-based methods**: GAT [35], GATv2 [4], SuperGAT [15], GlobalAttention [22], AGNN [33].
- **Pooling-based methods**: SortPool [48], DiffPool [43], Top-k Pool [10], SAGPool [19].
- **Kernel-based methods**: Graphlet kernel (GK) [31], Weisfeiler Lehman Kernel (WL) [30], Deep Graph kernels (DGK) [40].
- **GNN-based methods**: GCN [17], GIN [39]

Besides these methods, we also consider the state-of-the-art algorithms: IRM [2] and DRO [29], which are particularly designed for OOD issues. Please note that these methods require specific environments or group annotations for each training example, therefore we consider them as the methods with upper bound performance.

4.1.3 Hyper-parameters. All training hyper-parameters and model configurations are summarized in Appendix A.3.

4.2 Performance on Synthetic Graphs (RQ1)

To explore whether CAL can alleviate the OOD issue, we first conduct experiments on SYN-b with different biases: b ∈ {0, 0.2, ..., 0.9}. The experimental results are summarized in Table 1 and Figure 4. We have the following Observations:

**Obs 1: Refining discriminative features without considering the causality leads to poor OOD generalization.** For the unbiased dataset, most attention- and pooling-based baselines, such as GlobalAtt, SuperGAT, SortPool, Top-k Pool, outperform GCN. It indicates the effectiveness of extracting discriminative features in the ID setting. However, as the bias-level goes to extremes, the performance dramatically deteriorates. For instance, the performance drop of attention-based methods ranges from 7.37% ∼ 12.75% on SYN-0.1, and 3.79% ∼ 13.79% on SYN-0.9; Pooling-based methods drop from 7.82% ∼ 14.24% and 3.99% ∼ 12.10% for SYN-0.1 and SYN-0.9. These indicate that simply extracting discriminative features by attention or pooling module is prone to capture the data biases. These are also beneficial for reducing the training loss but lead to poor OOD generalization. Taking SYN-0.9 as an example, most "House" co-occur with "Tree" in the training data, so the model will mistakenly learn shortcut features from the "Tree"-type trivial subgraphs to make predictions, instead of probing the "House"-type causal subgraphs. This will mislead the model to adopt the "Tree" pattern to make decisions in the inference stage.

**Obs 2: GNNs with better ID performance tend to have worse OOD generalization.** For the unbiased dataset, GIN achieves the best performance (96.74%), while GAT (92.69%) outperforms the GCN (90.94%). This indicates that the in-distribution (ID) performance of these models exhibits such an order: GIN > GAT > GCN. However, when the bias is changed to 0.1 and 0.9, the performance of GIN drops by 9.55% and 7.36%, GAT drops by 8.71% and 5.47% and GCN drops by 6.60% and 5.43%, respectively. It shows that the rankings of models’ robustness against OOD issues are in the opposite order: GCN > GAT > GIN. This indicates that GNNs with better ID performance are prone to learn more shortcut features. Similar trends also occur in other baselines. After adopting the proposed CAL, this phenomenon is significantly alleviated, which verifies the effectiveness of CAL in overcoming the OOD issue.
Table 1: Test Accuracy (%) of graph classification on synthetic datasets with diverse biases. The number in brackets represents the performance degradation compared with the unbiased dataset. Our methods are highlighted with a gray background.

| Method            | SYN-0.1 | SYN-0.3 | Unbiased | SYN-0.7 | SYN-0.9 |
|-------------------|---------|---------|----------|---------|---------|
| GATv2 [4]         | 87.25 (7.37%) | 92.19 (2.12%) | 94.19 | 93.31 (0.93%) | 90.62 (3.79%) |
| SuperGAT [16]     | 83.81 (12.75%) | 91.94 (4.29%) | 96.06 | 88.50 (7.89%) | 82.81 (13.79%) |
| GlobalAtt [22]    | 87.19 (10.40%) | 93.75 (3.66%) | 97.31 | 94.62 (2.76%) | 91.50 (5.97%) |
| AGNN [33]         | 84.56 (11.69%) | 93.06 (2.81%) | 95.75 | 94.81 (0.98%) | 88.12 (7.97%) |
| DiffPool [43]     | 82.28 (8.69%) | 88.02 (2.32%) | 90.11 | 88.83 (1.42%) | 84.50 (6.23%) |
| SortPool [48]     | 80.70 (14.24%) | 92.33 (1.88%) | 94.10 | 92.14 (2.08%) | 90.35 (3.99%) |
| Top-k Pool [10]   | 84.31 (11.81%) | 93.53 (2.17%) | 95.60 | 94.44 (1.21%) | 88.02 (7.93%) |
| SAGPool [19]      | 88.08 (7.82%) | 90.86 (4.91%) | 95.55 | 92.22 (3.49%) | 83.99 (12.10%) |

| Method            | SYN-0.1 | SYN-0.3 | Unbiased | SYN-0.7 | SYN-0.9 |
|-------------------|---------|---------|----------|---------|---------|
| GCN [17]          | 84.94 (6.60%) | 89.38 (1.72%) | 90.94 | 90.25 (0.76%) | 86.00 (5.43%) |
| GCN + CAL         | 89.38 (6.03%) | 93.50 (1.70%) | 95.12 | 95.06 (0.06%) | 93.31 (1.90%) |
| GIN [39]          | 87.50 (9.55%) | 93.94 (2.89%) | 96.74 | 94.88 (1.92%) | 89.62 (7.36%) |
| GIN + CAL         | 93.19 (3.87%) | 96.31 (0.65%) | 96.94 | 96.56 (0.39%) | 95.25 (1.74%) |
| GAT [35]          | 84.62 (8.71%) | 89.50 (3.44%) | 92.69 | 92.31 (0.41%) | 87.62 (5.47%) |
| GAT + CAL         | 92.44 (4.37%) | 96.25 (0.42%) | 96.66 | 96.12 (0.56%) | 92.56 (4.24%) |

Table 2: Test Accuracy (%) of classification. For TUDataset, we perform 10-fold cross-validation and report the mean and standard derivations. Our methods are highlighted with gray background. If the performance improves, the number is bolded.

| Dataset | MUTAG | NCI1 | PROTEINS | COLLAB | IMDB-B | IMDB-M | MNIST | CIFAR-10 |
|---------|-------|------|----------|--------|--------|--------|-------|----------|
| GK [31] | 81.58±2.11 | 62.49±0.27 | 71.67±0.55 | 72.84±0.28 | 65.87±0.98 | 43.89±0.38 | -     | -        |
| WL [30] | 82.05±0.36 | 82.19±0.18 | 74.68±0.50 | 79.02±1.77 | 73.40±1.63 | 49.33±4.75 | -     | -        |
| DGK [40] | 87.44±2.72 | 80.31±0.46 | 75.68±0.54 | 73.09±0.25 | 66.96±0.56 | 44.55±0.52 | -     | -        |
| GlobalAtt [22] | 88.27±8.65 | 81.17±1.04 | 72.60±1.47 | 81.48±1.46 | 69.10±3.80 | 51.40±2.91 | -     | -        |
| AGNN [33] | 79.77±8.54 | 79.96±2.37 | 75.66±3.94 | 81.10±2.39 | 73.10±4.07 | 49.73±3.72 | -     | -        |
| DiffPool [43] | 85.61±6.22 | 75.06±3.66 | 76.25±4.21 | 79.24±1.66 | 74.47±3.84 | 49.20±3.10 | -     | -        |
| SortPool [48] | 86.17±7.53 | 79.00±1.68 | 75.48±1.62 | 77.84±1.22 | 73.00±3.59 | 49.53±2.29 | -     | -        |
| GCN [17] | 88.20±7.33 | 82.97±2.34 | 75.65±3.24 | 81.72±1.64 | 73.89±5.74 | 51.53±3.28 | 90.49 | 54.68    |
| GCN + CAL | 89.24±8.72 | 83.48±1.94 | 76.28±3.65 | 82.08±2.40 | 74.40±4.55 | 52.13±2.96 | 94.58 | 56.21    |
| GIN [39] | 89.42±7.49 | 82.71±1.52 | 76.21±1.83 | 82.08±1.51 | 73.40±3.78 | 51.53±2.97 | 96.51 | 56.36    |
| GIN + CAL | 89.91±8.34 | 83.89±1.93 | 76.92±3.31 | 82.68±1.25 | 74.13±5.21 | 52.60±3.26 | 96.93 | 56.63    |
| GAT [35] | 88.58±7.54 | 82.11±1.43 | 73.96±3.26 | 81.42±1.41 | 72.70±4.37 | 50.60±3.75 | 95.53 | 64.22    |
| GAT + CAL | 89.94±8.78 | 83.55±1.42 | 76.39±3.65 | 82.12±1.95 | 73.30±4.16 | 50.95±3.84 | 95.91 | 66.16    |

**Obs 3: Mitigating the confounding achieves more stable performance on OOD datasets.** We first define the performance discount on SYN-b as the accuracy on SYN-b normalized by the accuracy on unbiased SYN-0.5. It indicates the degree of the performance degradation on biased synthetic datasets, without considering the model’s ID generalization. We plot the performance discount curves on SYN-b with \( b \in \{0.1, 0.2, \ldots, 0.9\} \). As depicted in Figure 4, we observe that pooling-based methods outperform GIN in a small range of bias-levels (\( 0.2 - 0.8 \)), while the performance drops sharply when \( b = 0.1 \) or 0.9. For example, the performance discount of Top-k Pool drops from 0.95 to 0.88 as \( b \) reduces from 0.2 to 0.1. Attention-based methods perform worse than GIN when \( b < 0.5 \). For \( b > 0.5 \), AGNN achieves better performance than GIN, while GlobalAttention often performs worse. These results reflect that attention- or pooling-based methods all have their own weaknesses, such that they cannot consistently overcome the diverse distribution shifts. Equipped with CAL, GIN (red curve) consistently outperforms all the baselines on all ranges of bias-levels and obviously keeps a large gap, which further demonstrates the significance of mitigating the confounding effect, and the effectiveness of CAL. For comprehensive comparisons, we also plot two upper bound methods: IRM and DRO (dash lines), which require additional annotation information of trivial subgraphs for training. We observe that, even without additional information, CAL achieves comparable performance with these upper bound methods.

### 4.3 Performance on Real-world Graphs (RQ2)

Unlike synthetic graphs, there may not exist visible or specific patterns of the causal/trivial subgraphs in real-world graphs. However, there still exist irregular core-subgraphs [18, 23, 42, 45] that determine the predictions, which will inevitably involve different degrees of biases caused by the complementary parts. Similar to SYN-b, they mislead the GNNs to learn the spurious correlations.
Hence, we verify the practicability of CAL on eight real-world datasets. We report the results of the baselines from the original papers by default and reproduce the missing results. The results are summarized in Table 2 and we make the following Observations:

**Obs 4: The OOD issue is widespread in real-world datasets.** Attention-based and pooling-based methods are on a par with GNNs, and they both outperform graph kernel-based methods in most cases. It can be seen from the last six rows in Table 2, when CAL is applied to different GNN models, it consistently produces further performance improvements. It demonstrates that the distribution shifts also widely exist in real-world datasets. Specifically, we can find that GCN often performs worse than other GNNs, attention-based or pooling-based methods, while the performance significantly improves after adopting CAL. For instance, on IMDB-B and MNIST datasets, GCN+CAL achieves 1.92% and 4.52% relative improvements, respectively. This indicates that GCN is vulnerable to the distribution shift in certain datasets. Thanks to the causality, CAL will push GCN to pay more attention to causal features, which can establish robustness against the widespread OOD issues and achieve better generalization.

### 4.4 Ablation Study (RQ3)

In this section, we investigate the impact of the node/edge attention, random combination and the loss coefficients $\lambda_1$ and $\lambda_2$.

**Node Attention v.s. Edge Attention.** Node Attention (NA) and Edge Attention (EA) refine the features from two orthogonal views: node-level and edge-level. Here we want to examine the effect of adopting NA or EA alone. We adopt GCN as the encoder to conduct experiments on four biased synthetic datasets and two real-world datasets. GCN+CAL w/o NA or EA represents the node/edge attention scores in Equation (7)/(8) are evenly set as 0.5. The experimental results are shown in Figure 5. We can find that: (1) Comparing NA with EA, the performance of CAL without NA is significantly worse than that without EA, which indicates that the node feature contains more significant information compared with graph structure. (2) Just adopting NA or EA alone still achieves better performance than baselines, which demonstrates that only applying NA or EA can also disentangle the causal/trivial attended-graph and achieve causal intervention to some extent.

**Loss coefficients $\lambda_1$ and $\lambda_2$.** According to Equation (15), $\lambda_1$ denotes the strength of the disentanglement for the causal/trivial features, while $\lambda_2$ controls the strength of the causal intervention. To explore their impacts, we use GCN as the encoder and conduct experiments on two biased synthetic datasets and two real-world datasets. We fix one coefficient as 0.5 and change the other one in $\lambda_1$ and $\lambda_2$ with a step size of 0.1. The experimental results are shown in Figure 6. We can find that: (1) $\lambda_1$ achieves better performance in a range of $0.3 \sim 0.7$. Too small or too large values will cause performance degradation. (2) $\lambda_2$ is not as stable as $\lambda_1$. The optimal range is around $0.3 \sim 0.5$. It leads to a strong decline at $0.5 \sim 0.8$, which indicates that coefficient $\lambda_2$ should be set prudently.
4.5 Visualization and Analysis (RQ4)

Causal attended-graphs. We plot node/edge attention areas of the causal attended-graphs based on the attention scores in CAL. We adopt a GCN-based encoder and apply CAL on SYN-b and MNIST superpixel graphs. The visualizations are shown in Figure 7. Nodes with darker colors and edges with wider lines indicate higher attention scores. We surprisingly find that almost all the darker colors and wider lines precisely distribute on the deterministic areas, such as the causal subgraphs we defined in the synthetic dataset and the nodes located on digit pixels in MNIST superpixel graphs. It further demonstrates that the proposed CAL can effectively capture the causal features with insightful interpretations.

The explanation for performance improvements. Figure 8 displays the distribution of misclassification on SYN-b. The absissa represents the predictions, and the ordinate denotes the ground-truth types. The numbers in each row denote the proportion for each class. Figure 8 (Left) shows that the wrong predictions of graphs with “BA” are mainly distributed in “Cycle”, “Grid” and “Diamond” classes, while the wrong predictions of graphs with “Tree” mainly concentrate on the “House” class (highlighted by the red circle). On one hand, most of the “House” co-occur with “Tree” in the training data, GCN tends to capture the shortcut features, e.g., “Tree” patterns, to make decisions. Therefore, the other three causal subgraphs with “Tree” will mainly be classified as “House” in the testing set. On the other hand, only a few “House” causal subgraphs co-occur with “BA”, so the other three causal subgraphs with “BA” will almost not be misclassified as “House”. In contrast, Figure 8 (Right) shows that, by applying CAL, the concentration of misclassification is obviously alleviated. This demonstrates that CAL improves performance by mitigating the confounding effect.

5 RELATED WORK

Attention Mechanism selects the informative features from data, which has obtained great success in computer vision [7, 14, 36, 37, 41] and natural language processing tasks [6, 34]. In recent years, attention mechanism has gradually become prevalent in the GNN field. The attention modules for GNNs can be defined over edges [4, 16, 21, 33, 35] or over nodes [19, 20, 22]. Despite effectiveness, attention learning still stays at how to better fit the statistical correlations between data and labels. Hence, the learned attentions are inherently biased in OOD settings. Recent studies [36, 41] propose the causal attention modules to alleviate the bias. CaaM [36] adopts the adversarial training to generate the data partition in each iteration to achieve the causal intervention. CATT [41] proposes in-sample and cross-sample attentions based on front-door adjustment. However, they are both tailored for computer vision tasks, while cannot transfer to graph learning tasks, due to the irregular and challenging graph-structure data. Distinct from them, we utilize the disentanglement and causal intervention strategies to strengthen the attention modules for GNNs.

OOD Generalization [2, 13, 28, 29] has been extensively explored in recent years. IRM [2] minimizes the empirical risk under different environments. Group-DRO [29] adversarially explores the group with the worst risk and achieves generalization by minimizing the empirical risk of the worst group. Existing efforts [2, 28, 29] mainly focus on computer vision or natural language processing tasks, while the GNN field is of great need but largely unexplored. Furthermore, these methods require the environment or group prior information for each training sample, which is expensive in practice. To alleviate this dilemma, we adopt causal intervention to strengthen the causal relationship between the causal feature and prediction, thereby achieving better generalization.

Causal Inferences [26, 27] endows the model with the ability to pursue real causality. A growing number of studies [15, 25, 32, 47] have shown that causal inference is beneficial to diverse computer vision tasks. CONTA [47] uses backdoor adjustment to eliminate the confounder in weakly supervised semantic segmentation tasks. DDE [15] proposes to distill the colliding effect between the old and the new data to improve class-incremental learning. Unlike computer vision, the application of causal intervention in the GNN community is still in its infancy. CGI [9] explores how to select trustworthy neighbors for GNN in the inference stage, and demonstrates its effectiveness in node classification. Recent work [46] studies the connection between GNNs and SCM from a theoretical perspective. Different from them, we introduce a causal attention learning strategy to mitigate the confounding effect for GNNs. It encourages GNNs to pay more attention to causal features, which will enhance the robustness against the distribution shift.

Figure 7: Visualizations of causal attended-graphs. (Left): Synthetic graphs, (Right): MNIST superpixel graphs.

Figure 8: The misclassification distribution. Red circle highlights the concentration degree of misclassification.
6 CONCLUSION

In this work, we revisit the GNN modeling for graph classification from a causal perspective. We find that current GNN learning strategies are prone to exploit the shortcut features to support their predictions. However, the shortcut feature actually plays a confounder role. It establishes a backdoor path between the causal feature and the prediction, which misleads the GNNs to learn spurious correlations. To mitigate the confounding effect, we propose the causal attention learning (CAL) strategy for GNNs. CAL is guided by the backdoor adjustment from the causal theory. It encourages the GNNs to exploit causal features while ignoring the shortcut parts. Extensive experimental results and analyses verify its effectiveness. Future studies include adopting powerful disentanglement methods and more advanced causal intervention strategies to improve the CAL. We will also make efforts to apply CAL to other graph learning tasks, such as node classification or link prediction.

7 ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2020AAA0106000), and the National Natural Science Foundation of China (U19A2079, U21B2026). This research is also supported by CCCD Key Lab of Ministry of Culture and Tourism and Sea-NEXT Joint Lab.

REFERENCES

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE TPAMI 34, 11 (2012), 2274–2282.

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. arXiv preprint arXiv:1907.02893 (2019).

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. science 286, 5439 (1999), 509–512.

[4] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks? In ICLR.

[5] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shus- terman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heterocyclic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry 34, 2 (1991).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL. 4171–4186.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Moritz F. Hampel, Georg Heigold, Sylvus Gelly, et al. 2020. An Image is worth 16x16 Words: Transformers for Image Recognition at Scale. In ICLR.

[8] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Aidan N. Gomez. 2017. Attention is All You Need. In NeurIPS. 5998–6008.

[9] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? In ICLR.

[10] Elan Rosenfeld, Pradeep Kumar Ravikumar, and Christoph H. Lampert. 2020. The Risks of Invariant Risk Minimization. In ICLR.

[11] Asif Mirhoseini, Dejan Spasojevic, Nicole McKeown, and Dan Roth. 2018. Relating Graph Neural Networks to Structural Causal Models. arXiv preprint arXiv:2103.07035 (2018).

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In NeurIPS. 5998–6008.

[13] Petar Velivcković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In ICLR.

[14] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? In ICLR.

[15] Hua Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In IMLR. 1666–1674.

[16] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. Causal Intervention for Weakly-Supervised Semantic Segmentation. In NeurIPS. 430–438.

[17] Haoyuan Lan, Xia Hu, Jure Leskovec, and Stefanie Jegelka. 2021. On explainability of graph neural networks via subgraph explorations. In IMLR. 1666–1674.

[18] Jie Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In ICLR.

[19] Yunhui Lee, Inyop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In ICLR. 3734–3743.

[20] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. 2019. Understanding Attention and Generalization in Graph Neural Networks. In NeurIPS. 4204–4214.

[21] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2020. Shortcut learning in deep neural networks. Nature Machine Intelligence 2, 11 (2020), 655–673.

[22] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2020. Causal Intervention for Weakly-Supervised Semantic Segmentation. In NeurIPS. 1666–1674.

[23] John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. 2019. Graph convolutional networks with motif-based attention. In ICML. 499–508.

[24] Jie Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In ICLR.

[25] Chengyu Wang, Shang-Yuan Huang, Fang Sun, and Shuiwang Ji. 2019. Causal Atten- tion design with self-supervision. In CVPR.

[26] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? In ICLR.

[27] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explain- ability of graph neural networks via subgraph explorations. In IMLR. 1666–1674.

[28] Rong Wen. 2021. Counterfactual vqa: A cause-effect look at language bias. In CVPR. 12700–12710.

[29] Jiaxuan You, Keyulu Xu, and Jure Leskovec. 2021. Causal Attention for Interpretable and Generalizable Graph Classification. In KDD ’22, August 14–18, 2022, Washington, DC, USA.
A MORE IMPLEMENTATION DETAILS

A.1 Algorithm

We provide the detailed implementation of the proposed casual attention learning (CAL) in Algorithm 1. We adopt the causal attended-graph for prediction in the inference stage.

Algorithm 1: Casual Attention Learning

Input: Dataset $\mathcal{D}$, $f(\cdot)$, attention modules, classifiers, $\lambda_1$, $\lambda_2$.
Output: The trained parameters.
1. for sampled $M$ graphs $(G^k = (A^k, X^k))_{k=1}^M$ do
   2. for $k = 1$ to $M$ do
      3. $H^k \leftarrow f(A^k, X^k)$
      4. Compute $\alpha_{ci}, \alpha_t \leftarrow$ Equation (7) for all nodes of $G^k$
      5. Compute $\beta_{ci}, \beta_{hi} \leftarrow$ Equation (8) for all edges of $G^k$
      6. Get masks $\mathcal{M}^k_2$ and $\mathcal{M}^k_3$ based on $\beta_{ci}$ and $\alpha_{ci}$
      7. Get masks $\mathcal{M}^k_2$ and $\mathcal{M}^k_3$ based on $\beta_{hi}$ and $\alpha_{ti}$
      8. $G^{k}_c \leftarrow \{A^k \ominus \mathcal{M}^k_2, X^k \ominus \mathcal{M}^k_3\}$ // causal
      9. $G^{k}_t \leftarrow \{A^k \oplus \mathcal{M}^k_2, X^k \oplus \mathcal{M}^k_3\}$ // trivial
      10. $\mathbf{h}_{G^k_c}, \mathbf{z}_{G^k_c} \leftarrow$ Equation (9)
      11. $\mathbf{h}_{G^k_t}, \mathbf{z}_{G^k_t} \leftarrow$ Equation (10)
   end for
   13. Supervised loss: $L_{sup} \leftarrow$ Equation (11)
   14. Uniform loss: $L_{unif} \leftarrow$ Equation (12)
   15. $I \leftarrow$ Shuffle($\{1, \ldots, M\}$)
   16. for $k = 1$ to $M$ do
      17. $i \leftarrow I[k]$
      18. $\mathbf{h}_{G^i_k} \leftarrow \mathbf{h}_{G^i_c} + \mathbf{h}_{G^i_t} // \text{random combination}$
      19. $\mathbf{z}_{G^i_k} \leftarrow \Phi(\mathbf{h}_{G^i_k})$
   end for
   21. Causal loss: $L_{caus} \leftarrow$ Equation (14)
   22. Total loss: $L \leftarrow L_{sup} + \lambda_1 L_{unif} + \lambda_2 L_{caus}$
   23. Update all the trainable parameters to minimize $L$
   end for

A.2 Datasets Details

In this section, we give more details about the synthetic datasets and real-world datasets.

1) Synthetic graphs. For each synthetic graph instance, it consists of two subgraphs: trivial and critical subgraphs. We introduce the proposed trivial subgraph and critical subgraph as follows:

- **Trivial subgraph.** There exist two types of trivial subgraphs: BA-SHAPES and Tree. The BA-SHAPES is a Barabási-Albert (BA) graph [3], and we abbreviate it as “BA” in this paper. The “Tree” graph is a base 12-level balanced binary tree [42]. To reduce the influence, we control the number of nodes in the two kinds of trivial subgraphs to be similar.

- **Causal subgraph.** There are four types of causal subgraphs: “House”, “Cycle”, “Grid”, “Diamond”. The visualizations of these trivial subgraphs and causal subgraphs are depicted in Figure 3.

For each synthetic graph instance, a causal subgraph is randomly attached on one node of a trivial subgraph. Then the resulting graph is further perturbed by adding 10% random edges. We take the one-hot form of the node degree as the node feature and set the dimension of node feature to 20. The synthetic graph examples are displayed in Figure 3. The statistics of the synthetic datasets are summarized in Table 3. We split the dataset into training, validation and testing set with the ratio of 7:1:2.

| Dataset       | #Graphs | #Nodes | #Edges | #Classes |
|---------------|---------|--------|--------|----------|
| SYN-b         | 8000    | 230~247| 542~1000 | 4        |
| MUTAG         | 188     | 17.93  | 19.79  | 2        |
| NCI1          | 4110    | 29.87  | 32.30  | 2        |
| PROTEINS      | 1113    | 39.06  | 72.82  | 2        |
| COLLAB        | 5000    | 74.49  | 2457.78 | 3        |
| IMDB-B        | 1000    | 19.77  | 96.53  | 2        |
| IMDB-M        | 1500    | 13.00  | 65.94  | 3        |
| MNIST         | 70000   | 70.57  | 564.66 | 10       |
| CIFAR-10      | 60000   | 117.63 | 941.04 | 10       |

2) Real-world graphs. To demonstrate the practicality of the proposed CAL, we conduct experiments on TUDataset [24] and Superpixel graphs [18]. For TUDataset, we gather three biological datasets (MUTAG, NCI1, PROTEINS) and three social networks datasets (COLLAB, IMDB-B, IMDB-M), which are commonly used in graph classification benchmarks [8, 39]. Following [8, 39, 43], we use 10-fold cross-validation and report average accuracy and standard deviation. The superpixel graphs [8, 18] includes MNIST and CIFAR-10, which are classical image classification datasets converted into graphs using superpixels technology [1] and assigning each node’s features as the superpixel coordinates and intensity. Following [8, 18], we split the MNIST and CIFAR-10 to 55K training/5K validation/10K testing, and 45K training/5K validation/10K testing, respectively. All the detailed statistics about the real-world datasets are summarized in Table 3.

A.3 Hyper-parameters

As for training parameters, we train the models for 100 epochs with batch size of 128. We optimize all models with the Adam optimizer and default choice of learning rate (i.e., 0.001) for all experiments. For SYN-b and TUDataset, we use GCN, GIN and GAT as GNN encoders with 3 layers and 128 hidden units. For Superpixel graphs MNIST and CIFAR-10, we use the GNN encoders with 4 layers and 146 hidden units as [8]. For all the baselines, we follow the default settings from original papers and reproduce the missing results. For the proposed CAL, we search $\lambda_1$ and $\lambda_2$ in (0.1, 1.0) with a step size of 0.1 and report the results with the best settings. We adopt NVIDIA 2080 Ti (11GB GPU) to conduct all our experiments, the training time comparison is shown as Table 4.

| Method       | SYN-b | MUTAG | NCI1 | IMDB-M | MNIST |
|--------------|-------|-------|------|--------|-------|
| GCN          | 4.16  | 1.03  | 12.71| 4.61   | 57.20 |
| GCN + CAL    | 6.67  | 1.35  | 17.37| 6.16   | 75.80 |