Violin: Virtual Overbridge Linking for Enhancing Semi-supervised Learning on Graphs with Limited Labels

Siyue Xie , Da Sun Handason Tam , Wing Cheong Lau
The Chinese University of Hong Kong
{xs019, tds019, wclau}@ie.cuhk.edu.hk,

Abstract

Graph Neural Networks (GNNs) is a family of promising tools for graph semi-supervised learning. However, in training, most existing GNNs rely heavily on a large amount of labeled data, which is rare in real-world scenarios. Unlabeled data with useful information are usually underexploited, which limits the representation power of GNNs. To handle these problems, we propose **Vi**rtual **O**verbridge **Lin**king (Violin), a generic framework to enhance the learning capacity of common GNNs. By learning to add virtual overbridges between two nodes that are estimated to be semantic-consistent, labeled and unlabeled data can be correlated. Supervised information can be well utilized in training while simultaneously inducing the model to learn from unlabeled data. Discriminative relation patterns extracted from unlabeled nodes can also be shared with other nodes even if they are remote from each other. Motivated by recent advances in data augmentations, we additionally integrate Violin with the consistency regularized training. Such a scheme yields node representations with better robustness, which significantly enhances a GNN. Violin can be readily extended to a wide range of GNNs without introducing additional learnable parameters. Extensive experiments on six datasets demonstrate that our method is effective and robust under low-label rate scenarios, where Violin can boost some GNNs' performance by over 10% on node classifications.

1 Introduction

Graph Neural Networks (GNNs) are deemed to be powerful tools to analyze graph-structured data in recent researches, where successful applications are found in tasks such as traffic predictions [Li *et al.*, 2017], recommendations [Li *et al.*, 2020], protein analysis [Gligorijević *et al.*, 2021], etc. In general, a GNN learns knowledge for target nodes by aggregating information from neighbors. By stacking multiple layers, topological features and mutual relationships can be embedded into a node representation. In semi-supervised settings, by pairing the node representations with their labels, typical GNNs [Kipf and Welling, 2016; Veličković *et al.*, 2018] can be trained through backpropagation. Learned representations can therefore be used for downstream tasks such as node classifications or link predictions.

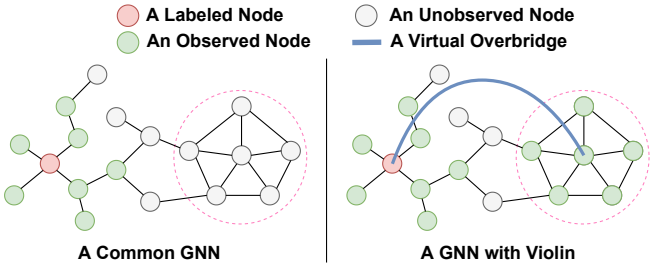

Figure 1: An example about a two-layer GNN. Suppose there is only a labeled node in the graph. ***Left:*** For a common GNN, during training, the receptive field only covers the labeled (target) node and its two-hop neighbors, while leaving others in the graph unobserved. Some useful but critical information, e.g., a star-shaped hub structure marked by the pink dot circle, is not perceived. ***Right:*** A GNN enhanced by the proposed Violin framework. The enlarged receptive field enables the model to incorporate more unlabeled nodes into training and therefore captures more patterns in the graph.

However, in real-world scenarios of semi-supervised learning, labeled entities are usually the minority since it is expensive to annotate an instance in a graph. It can be the Achilles' heel of a model because most GNNs rely heavily on labeled data for training. Such a discrepancy may degrade the representation power of a GNN and prevent it from generalizing to more practical applications. In addition, many unlabeled instances are usually ignored in training since they are remote from any labeled node, which makes them unobserved in the aggregation process. (An observed/ unobserved node is a node inside/ outside the receptive field during training.) An example on a two-layer GNN is shown in the ***Left*** of Figure 1. The drawbacks are obvious: some useful or critical patterns among unlabeled nodes, e.g., the star-shaped structure marked by the pink dot circle in Figure 1, are not captured, resulting in learning inferior representations. The model can also be biased by the limited labeled data, which may lead to more serious problems about ethics and fairness if applied to real-world tasks.

To tackle the aforementioned problems, we propose **Vi**rtual **O**verbridge **Lin**king (Violin), a generic framework to enhance the learning capacity of typical GNN models for graph semi-supervised learning with limited labels. Instead of focusing only on labeled instances, we jointly incorporate both the labeled and unlabeled data into learning. As shown in the ***Right*** of Figure 1, we generate a new graph by adding virtual overbridges (VOs) between a pair of nodes, which are estimated to be highly-correlated through an adaptive thresholding mechanism. Supervised signals from labeled nodes can be utilized as auxiliary information to generate embeddings for unlabeled nodes, while specific patterns that only appear in unlabeled data can be propagated to labeled nodes and captured by the model. In this way, we effectively correlate both the labeled and unlabeled nodes. Learned knowledge can be shared within the same class of nodes, even if they are remote from each other in the original graph. Since VOs selectively extend the receptive field of a GNN towards informative regions, aggregations with VOs can be more discriminative to task-specific patterns, which substantially improves the representation ability of GNNs. In addition, the new graph derived by Violin can be viewed as an augmentation of the original graph. Motivated by previous works on graph data augmentations, we naturally integrate Violin with the consistency regularized training [Veličković *et al.*, 2018]. Unlabeled data are efficiently incorporated into the learning process, which induces the model to yield node representations with better robustness and further enhances the representation power of a GNN. Instead of designing more sophisticated network architectures, we enhance GNNs' representation power by learning from all available data appropriately. Thus, Violin can be readily extended to a wide range of GNNs without introducing additional learnable parameters. Extensive experiments on node classification tasks demonstrate the superiority and effectiveness of Violin over canonical GNN models and other state-of-the-art enhancement techniques. Codes and the appendix are available at *https://github.com/xslangley/violin*. Our contributions can be summarized as follows:

1. We propose Violin, a generic enhancement scheme for GNNs on semi-supervised learning. By learning to add VOs, labeled and unlabeled data are proactively correlated. This improves information propagations and helps extract discriminative patterns, which facilitates the learning capacity of a GNN.
2. We substantially investigate the applicable conditions/ 'sweet spots' of Violin (on **Appendix C**) and empirically verify that Violin is effective with different number of VOs and robust to noisy estimations in a wide range.
3. We build a comprehensive framework by integrating Violin with consistency regularized training and an adaptive threshold mechanism. By taking all available nodes into account, we train a GNN with better data utilization, which yields more powerful and robust representations.
4. Compared with previous methods, Violin can be readily applied to a wide range of GNN models with only little computational overheads. Qualitative and quantitative experiments demonstrate the effectiveness of Violin over other state-of-the-art methods.

2 Related Works

2.1 GNNs for Graph Semi-supervised Learning

GNNs are widely used for graph semi-supervised learning in the recent studies. Some works concentrate on learning powerful representations with specific architectures. For example, the attention mechanism [Veličković *et al.*, 2017; Zhang *et al.*, 2018] is introduced to specify the importance of neighbors during aggregation. GDC [Klicpera *et al.*, 2019] and APPNP [Klicpera *et al.*, 2018] leverage graph diffusion to derive a better scheme for message propagation. Some other methods [Xu *et al.*, 2018a; Xu *et al.*, 2018b; Corso *et al.*, 2020; Chen *et al.*, 2020b] aim at designing strictly and theoretically more expressive operators for GNNs. All these works effectively promote the basic capability of GNN to learn from graph-structured data.

2.2 Learning-Enhancement Techniques for GNNs

In semi-supervised scenarios, a GNN may only extract limited knowledge from a graph due to few labeled data, even though it has the potential to learn powerful representations. To handle this problem, some researchers [Sun *et al.*, 2020; Li *et al.*, 2018] resort to self-training, where the labels set is progressively extended by including unlabeled instances during training. There are also works [Huang *et al.*, 2020; Klicpera *et al.*, 2018] that enhance GNNs by disentangling the trainable prediction model from the message passing framework. Each part can be tuned flexibly, which achieves lower computational cost. Some works pretrain a GNN based on unsupervised/ self-supervised mechanisms by graph augmentations [Sun *et al.*, 2019; Xu *et al.*, 2021; You *et al.*, 2021; You *et al.*, 2020; Zhu *et al.*, 2021; Wan *et al.*, 2020; Thakoor *et al.*, 2021; Hassani and Khasahmadi, 2020]. Handcrafted graph augmentations can therefore be introduced to assist graph semi-supervised learning [Feng *et al.*, 2020; Xie *et al.*, 2022; Zhao *et al.*, 2021; Chen *et al.*, 2020a]. Different from previous methods, the augmentation in Violin, i.e., virtual overbridges, are fully data-driven (instead of relying on human knowledge). Unlike self-training, unlabeled instances in Violin are utilized to assist supervised learning more appropriately without polluting the original labeled/ training set. Violin can also be trained end-to-end (instead of a multi-stage manner), which is more efficient and yields task-oriented representations. Detailed comparisons refer to **Appendix A**.

3 Motivations and Preliminary Evaluations

3.1 Notations and Problem Formulations

In this paper, we define a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V} = \{v_i | i = 1, 2, ..., N\}$ is the node set, $\mathcal{E}$ is the edge set and $\boldsymbol{X} \in \mathbb{R}^{N \times d_{in}}$ is the node feature matrix. In semi-supervised settings, $\mathcal{V}$ can be further separated as $\mathcal{V} = \{\mathcal{V}^L, \mathcal{V}^U\}$, where $\mathcal{V}^L$ and $\mathcal{V}^U$ are the labeled and unlabeled set respectively. The labels set $\boldsymbol{Y}^L = \{\boldsymbol{y}_v | v \in \mathcal{V}^L\}$, where $\boldsymbol{y}_v \in \{0, 1\}^K$ is represented by a one-hot vector indicating one of the K classes. A general GNN model is to learn a mapping that takes the graph $\mathcal{G}$ as input and generate representations for each node: $\boldsymbol{H} = GNN(\mathcal{G}) = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N]^\intercal \in \mathbb{R}^{N \times d_h}$.

Depth	Cora (5.17%)	Citeseer (3.62%)	Pubmed (0.30%)
1	23.78	14.29	1.80
2	61.45	32.82	14.19
3	81.91	49.68	46.97

Table 1: The ratio (%) of observed nodes with regard to the depths (layers) of a GNN (label rate in parentheses).

In this paper, our goal is to learn a GNN to predict the class label for each node, given the graph and limited labels.

3.2 Rethinking Graph Semi-supervised Learning Paradigm

In general, we can train a GNN through backpropagation by measuring the prediction errors on the labeled nodes. For example, the loss for node classifications is usually measured by the cross-entropy:

$$\mathcal{L}_{ori} = -\frac{1}{|\mathcal{V}^L|} \sum_{v \in \mathcal{V}^L} \boldsymbol{y}_v^\intercal \log\left(\boldsymbol{p}_v\right), \ \boldsymbol{p}_v = Softmax(\boldsymbol{h}_v) \quad (1)$$

where $\boldsymbol{p}_v$ indicates the confidence vector for node v.

However, in real-world cases, annotation is expensive and sometimes even difficult for domain experts, which means labeled data are usually scarce in a graph. Therefore, following the general training paradigm, only a small part of data can be utilized by GNNs, while keeping the majority idle in the learning process. As shown in the ***Left*** of Figure 1, with limited labels, the receptive field of a two-layer GNN only covers the neighbors within two-hops of the labeled node. In other words, most of the data are under-exploited. Such a GNN only learns limited knowledge from the graph and fail to capture some important but critical patterns that only appear in those unobserved unlabeled data. The learning capacity of a GNN is therefore suppressed, which results in generating less powerful representations.

To quantify the amount of data under-exploited, we list the ratio of observed nodes (during training) of three recognized datasets with different depths (number of layers) of a typical GNN, e.g., GCN [Kipf and Welling, 2016]. As shown in Table 1, the data utilization is at a relatively low level. Even if we stack three layers to enlarge the receptive field, there are still more than half of the data in Citeseer and Pubmed untouched during training. We can stack a deeper model until the receptive field covers all available data. However, recent studies reveal that GNNs suffer from over-smoothing problem in deep layers [Li *et al.*, 2018; Wu *et al.*, 2020; Rong *et al.*, 2019]. Even if introducing the attention mechanism, which is recognized to highlight important neighbors, can hardly help. We empirically demonstrate it in **Appendix B**. In addition, the number of neighbors grows exponentially as the depth/ receptive field goes further away from the target node. Discriminative information will be attenuated by noisy signals during aggregation. Empirical experiments [Kipf and Welling, 2016; Veličković *et al.*, 2017; Wu *et al.*, 2019] also show that shallow GNNs performs better in few-labeled graphs. Therefore, instead of stacking deeper GNNs, we should explore other ways to better make use of those unlabeled but available data.

3.3 Enhancement via Virtual Overbridges Linking

Following the above analyses, we can observe that the learning capacity of a GNN model can be restricted by the data utilization. Therefore, to enhance a GNN model, a natural next-step is to learn from the rich unlabeled data.

A naive way to incorporate more data into the learning process is to extend the receptive field of GNNs. However, enlarging the receptive field omnidirectionally may result in performance collapse because of over-smoothing. Instead of stacking a deeper model, our idea is to learn to selectively add **V**irtual **O**verbridges (VOs), which act as special edges to propagate messages in a graph, within the same class of nodes. We therefore term this scheme as **Vi**rtual **O**verbridge **Lin**king (Violin). There are many potential benefits that Violin can bring to a GNN. On the one hand, VOs naturally extend the receptive field of a GNN in training, as shown in the ***Right*** of Figure 1. Different from simply stacking more layers, Violin specifically extends the receptive field towards the regions related to the target node. This encourages a GNN to collect class-specific features along VOs, which results in yielding task-related node representations. On the other hand, diverse knowledge learned from different nodes can be shared with other physically-remote instances. Since a graph is an irregular data structure with complex relationships, the contexts (i.e., the local neighborhood information of a node including the node/ edge features and the graph topology) of nodes vary greatly even if they come from the same class. By adding VOs, the patterns and knowledge learned from different contexts can be propagated to others, encouraging the model to generate a more comprehensive and robust representation that fits the target node and its role.

To verify the enhancement potential of Violin for graph semi-supervised learning, we conduct a series of experiments with an ideal setting. Suppose we know the oracles, i.e., the class labels, of all nodes in the graph. Let $\mathcal{V}_k$ be the node set of the k-th class. The set of all candidate VOs can be formulated as follows:

$$\mathcal{E}_{vo} = \{(u, v) | u, v \in \mathcal{V}_k, \ k = 1, 2, ..., K\} \quad (2)$$

where (u, v) is an VO between node u and v. For each node, we randomly pick m other nodes within the same class to build VOs. These operations are the same as **Step 2** in Figure 2. We denote the set of selected VOs as $\tilde{\mathcal{E}}_{vo} \subset \mathcal{E}_{vo}$ and the new graph after adding VOs as $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}}, \boldsymbol{X})$, where $\tilde{\mathcal{E}} = \mathcal{E} \cup \tilde{\mathcal{E}}_{vo}$. Since VOs only act as an information pathway to propagate messages from peers, the graph is expected to keep the same semantics, i.e., the role of each node is kept the same. We therefore name $\tilde{\mathcal{G}}$ as 'semantics-consistent graph'. Compared with the original graph, $\tilde{\mathcal{G}}$ is more flexible on information propagation. We then train and assess a GCN model on $\tilde{\mathcal{G}}$ to investigate its performance for node classifications. All experimental settings follow the work of GCN [Kipf and Welling, 2016], except that we replace $\mathcal{G}$ with $\tilde{\mathcal{G}}$. Note that the semantics-consistent graph is not unique as we can generate a new $\tilde{\mathcal{G}}$ for each training epoch. The GCN trained on $\tilde{\mathcal{G}}$ with oracles is denoted as GCN-O$_{(m,r)}$, where m is the number of added VOs per node and r is the proportion of noisy (incorrect) VOs we deliberately add for robustness test.

Model	Cora	Citeseer	Pubmed
GCN	82.52	71.02	79.16
GCN-O$_{(1,0.3)}$	90.2 (+7.68)	81.64 (+10.62)	84.25 (+5.09)
GCN-O$_{(1,0)}$	95.68 (+13.16)	89.41 (+18.39)	93.07 (+13.91)

Table 2: Node classifications (%) of Violin with oracles and noises.

Results in Table 2 show a striking improvement over GCN by adding only one VO per node, even if up to 30% of added VOs are noisy ($r = 0.3$). With the new graph ($\tilde{\mathcal{G}}$) created by Violin, GCN-O has more chances to access nodes that used to be unobserved. Specific patterns of unobserved data are now more likely to be captured. As for GCN-O$_{(1,0)}$, messages propagated through VOs are purely class-related features without noises, which are highly correlated with the target node. Such features are informative in the learning process and more effective when more correct VOs are added (refer to **Appendix C.1**). Furthermore, we investigate the 'sweet spot' of Violin with a series of harsher settings and empirically show in **Appendix C.2** that Violin affords up to 70% of noisy VOs and still works if limited VOs are added. All these observations show the promising power of Violin.

It should be noted that all above experiments are conducted under ideal settings, i.e., we know the oracles for adding VOs, which is unfair to compare with other works. However, we will show in **Section 5** with fully comparable experimental settings on six real-world datasets that Violin substantially enhances various GNNs and outperforms other works.

4 Violin for Graph Semi-supervised Learning

In this section, we extend Violin to practical scenarios of graph semi-supervised learning. An overview of Violin's pipeline is shown in Figure 2.

4.1 Label Estimations for Ignition

In preliminary experiments, we generate semantics-consistent graphs via oracles. However, in real-world scenarios, labels are unavailable for most nodes. To handle this problem, we propose to use estimated labels to substitute oracles. Although it can be noisy, it is demonstrated to be effective by previous self-training works [Sun *et al.*, 2020; Li *et al.*, 2018]. We have also shown in Table 2 and **Appendix C.2** that Violin is robust to wrong VOs due to noisy estimations. Specifically, we train a vanilla GNN on the few-labeled graph following the basic learning paradigm (Equation 1). The vanilla GNN model can therefore predict the class of each unlabeled nodes for Violin to ignite. As shown in the **Step 1** of Figure 2, we initialize Violin by associating each node with its predicted class label. The graph with predicted labels is termed as label-estimated graph for clarity. Note that the ground-truth label of training nodes are given in advance. We use their true labels instead of their predictions for the label-estimated graph.

4.2 Violin with Adaptive Confidence Threshold

With estimated labels, we can construct semantic-consistent graphs for Violin. However, the great performance gain shown in Table 2 relies on oracles, where most VOs are correct and unlikely to introduce noises. In contrast, wrong predictions inevitably exist in the estimated labels. The 'correctness' of an VO will be challenged if any of its two end-points uses untrusted predicted labels. For example, the prediction accuracy of GCN in the Citeseer dataset is around 70%. Then the ratio of VOs that both end-points are correctly predicted is expected to be 49% (0.7×0.7). In other words, more than half of VOs may propagate undesired information, which may hurt the performance in the worst case.

An intuitive solution for this problem is to selectively pick trusted nodes for Violin, which reduces the number of incorrect VOs. Recall that in Equation 1, each predicted label is associated with a confidence value. We can therefore set a confidence threshold for nodes to refine the VO set based on the prediction results in the **validation set**. Figure 3 shows an example of the confidence distribution in Cora. We can empirically set a static threshold, for example, by 0.7, where the expected prediction accuracy of remaining (qualified) nodes are perfect (100%).

However, as shown in **Appendix E.4**, the confidence distribution will shift in training, where a static threshold cannot always fit. Our experiments shown in **Appendix C.2** also suggest making a balance between the quality (correctness) and quantity of added VOs. Thus, we propose to adaptively adjust the confidence threshold during training. The key idea is to always keep a good enough (expected) accuracy for the qualified nodes (after thresholding) to generate high-quality VOs and simultaneously add enough VOs to the graph. Specifically, let $\rho \in (0, 1)$ be a candidate confidence threshold. The qualified node set based on ρ can be formulated as follows:

$$\mathcal{V}_\rho = \{v|\phi_v > \rho,\ v \in \mathcal{V}\} \tag{3}$$

where ϕ_v is the confidence value of the predicted label of node v. The nodes set of the **validation set** are denoted by $\hat{\mathcal{V}}$. The nodes in $\hat{\mathcal{V}}$ that fulfill the threshold are:

$$\hat{\mathcal{V}}_\rho = \{v|\phi_v > \rho,\ v \in \hat{\mathcal{V}}\},\quad \hat{\mathcal{V}}_\rho = \hat{\mathcal{V}}^c_\rho \cup \hat{\mathcal{V}}^w_\rho \tag{4}$$

where $\hat{\mathcal{V}}^c_\rho$ and $\hat{\mathcal{V}}^w_\rho$ are the set of correctly and wrongly predicted nodes in $\hat{\mathcal{V}}$. We then derive the optimal threshold by:

$$\rho^* = \arg\min_\rho \{\ \rho \mid\ |\hat{\mathcal{V}}^c_\rho|/|\hat{\mathcal{V}}_\rho| \geq \delta\} \tag{5}$$

where $\delta \in (0, 1)$ is the expected prediction accuracy (set manually) of qualified nodes and $|\cdot|$ denotes the cardinality. With ρ^*, we only add VOs for the nodes in $\mathcal{V}_{\rho^*}$, as **Step 2** of Figure 2.

Based on Equation 5, ρ^* changes dynamically in each training epoch. Compared with the static thresholding, such a scheme fits the shifting confidence distribution adaptively, which helps Violin create high-quality VOs for the graph. The added VOs encourage the model to collect class-related information from their remote peers, resulting in generating more discriminative representations for nodes.

In addition, we regularize Violin by the representations of the two end-points of added VOs. Since an VO is expected to link two nodes within the same class, the representations

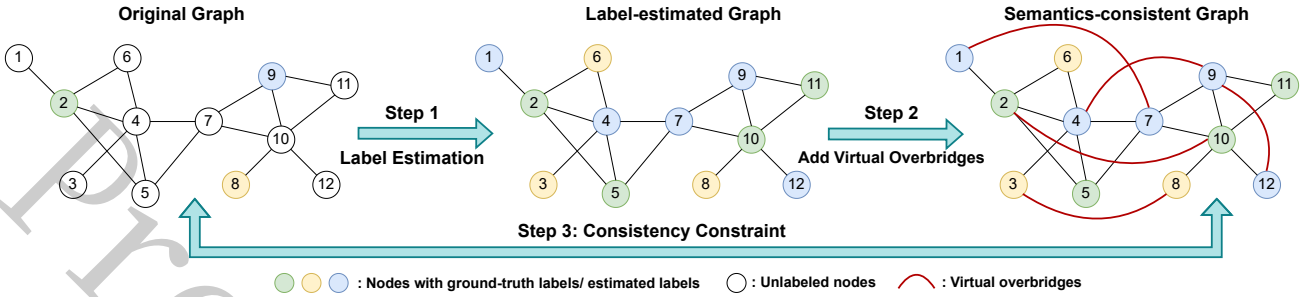

Figure 2: The pipeline of Violin. We first train a vanilla GNN model on the few-labeled graph to estimate the class of all unlabeled nodes. With the label-estimated graph, VOs can be selectively added to a pair of nodes that are predicted to be in the same class. The node representations learned from the semantics-consistent graph are forced to keep consistent with that of the original graph.

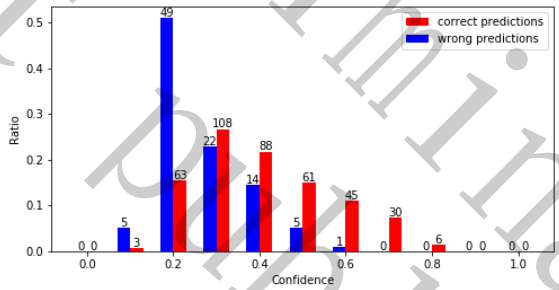

Figure 3: An example: the confidence distribution of the validation set on the Cora dataset by GCN. The number of samples in each interval is shown in the top of each bar.

of its two end-points should align with each other. Specifically, let $\tilde{\boldsymbol{H}}$ be the node representations generated based on the semantic-consistent graph $\tilde{\mathcal{G}}$:

$$\tilde{\boldsymbol{H}} = GNN(\tilde{\mathcal{G}}) = [\tilde{\boldsymbol{h}}_1, \tilde{\boldsymbol{h}}_2, ..., \tilde{\boldsymbol{h}}_N]^\intercal \in \mathbb{R}^{N \times d_h} \quad (6)$$

where d_h is the dimension of node representations. Then the VO loss can be defined as follows:

$$\mathcal{L}_{vo} = \frac{1}{|\tilde{\mathcal{E}}_{vo}|} \sum_{(u,v)} |\tilde{\boldsymbol{h}}_v - \tilde{\boldsymbol{h}}_u|_1, \quad for \ \ (u,v) \in \tilde{\mathcal{E}}_{vo} \quad (7)$$

$\mathcal{L}_{vo}$ forces a GNN to keep consistency between the two end-points, inducing Violin to generate more reasonable VOs during training. On the other hand, such a regularization term prevents the model from overfitting to specific nodes, which helps yield more generalized representations.

4.3 Augmentations by Semantics Consistency

From the view of data augmentation, the semantic-consistent graph $\tilde{\mathcal{G}}$ is equivalent to an augmentation of the original graph. The semantics of a node in $\tilde{\mathcal{G}}$ should be consistent with that of it in $\mathcal{G}$. Following the work of [Feng *et al.*, 2020], we additionally adopt consistency regularization training for Violin to keep the semantics consistency. Specifically, we denote the representations of a node v learned from $\mathcal{G}$ and $\tilde{\mathcal{G}}$ as $\boldsymbol{h}_v$ and $\tilde{\boldsymbol{h}}_v$, respectively. We then define the consistency loss on the unlabeled nodes by the L1-norm as follows:

$$\mathcal{L}_{con} = \frac{1}{|\mathcal{V}^U|} \sum_{v} |\boldsymbol{h}_v - \tilde{\boldsymbol{h}}_v|_1, \quad v \in \mathcal{V}^U \quad (8)$$

By introducing the consistency loss, Violin purposefully incorporates all unlabeled nodes into training. Compared with GNNs that only cover instances within the receptive filed of labeled nodes, Violin is able to learn knowledges from all nodes. Augmentations on graph topology also encourage Violin to yield representations with better robustness.

4.4 Model Training for Violin

The overall loss of Violin for training is formulated as:

$$\mathcal{L} = \mathcal{L}_{cls} + \gamma \mathcal{L}_{vo} + \alpha \mathcal{L}_{con} \quad (9)$$

where γ and α are hyperparameters to balance the learning. $\mathcal{L}_{cls}$ is the classification loss derived from labeled nodes. To better utilize the supervised signals from both the original and semantics-consistent graph, we define $\mathcal{L}_{cls}$ as follows:

$$\mathcal{L}_{cls} = -\frac{1}{2|\mathcal{V}^L|} \left(\sum_{v \in \mathcal{V}^L} \boldsymbol{y}_v^\intercal \log(\boldsymbol{p}_v) + \sum_{v \in \mathcal{V}^L} \boldsymbol{y}_v^\intercal \log(\tilde{\boldsymbol{p}}_v) \right) \quad (10)$$

where $\tilde{\boldsymbol{p}}_v$ is the prediction confidence vector of node v in the semantics-consistent graph, as similar to $\boldsymbol{p}_v$.

In training, we reuse the vanilla model at **Step 1** of Figure 2 as the backbone and continue to find-tune it by Equation 9, which effectively improves the learning efficiency. Estimated labels for Violin can also be updated at each epoch by the prediction results of the fine-tuned model, which enables the Violin to generate more appropriate VOs as the training goes.

5 Experimental Evaluations

5.1 Experimental Setups

Datasets

We evaluate Violin by node classifications on six public datasets: Cora, Citeseer, Pubmed, Ogbn-arxiv (Ogbna), Amazon-Photo (AmzP) and Coauthor-CS (CoCS), where the first four have publicly-recognized or official splits [Kipf and Welling, 2016; Wu *et al.*, 2019; Hu *et al.*, 2020]. Statistical details refer to **Appendix D.1** in the supplementary.

Model	Cora	Citeseer	Pubmed	Ogbn-arxiv
†MLP	58.51±0.80	55.64±0.46	72.71±0.61	55.50±0.23
†n2v	72.35±1.41	50.82±0.96	62.03±1.05	70.07±0.13
†DGI	82.30±0.60	71.80±0.70	76.80±0.60	N/A
†MVGRL	82.90±0.70	72.60±0.70	79.40±0.30	N/A
†DIMP	83.30±0.50	73.30±0.50	81.40±0.50	N/A
PNA	75.05±3.37	55.04±4.85	75.91±1.17	69.54±0.58
GIN	78.83±1.45	66.87±0.96	77.83±0.42	63.19±1.57
JK-Net	80.35±0.58	67.29±1.02	78.36±0.31	72.19±0.24
SGC	80.70±0.55	71.94±0.07	78.82±0.04	68.59±0.03
SAGE	81.73±0.58	69.86±0.62	77.20±0.40	72.04±0.20
GCN	82.52±0.60	71.02±0.83	79.16±0.35	71.99±0.22
GAT	82.76±0.88	71.87±0.53	77.74±0.34	71.75±0.28
APPNP	83.13±0.58	71.39±0.68	80.30±0.17	71.22±0.26
GCNII	84.17±0.40	72.46±0.74	79.85±0.34	72.46±0.32
†AdaEdge($GCN_{N/A}$)	82.30±0.80	69.10±0.90	77.40±0.50	N/A
†CG^3	83.40±0.70	73.60±0.80	80.20±0.80	N/A
†GAUG-O(GCN_{128})	83.60±0.50	73.30±1.10	79.30±0.40	71.40±0.50
†CoCoS($GCN_{16/256}$)	84.15±N/A	73.57±N/A	80.92±N/A	71.77±N/A
†GRAND(GCN_{32})	84.50±0.30	74.20±0.30	80.00±0.30	N/A
†GAM(GCN_{128})	84.80±0.06	72.46±0.44	81.00±0.09	N/A
Violin(GCN_{16})	**85.22**±0.60	73.38±0.32	81.11±0.47	-
Violin(GCN_{32})	85.08±0.65	73.96±0.38	81.05±0.51	-
Violin(GCN_{128})	84.49±0.66	**74.26**±0.40	**81.23**±0.42	-
Violin(GCN_{256})	84.03±0.59	74.16±0.55	80.83±0.36	**72.49**±0.09

Table 3: Node Classifications Accuracy (%). ('†': results are collected from published papers/ leaderboards. 'N/A': the data is not reported. The subscript of a model: the hidden layer dimensions.)

Models for Comparisons

We compare Violin with twenty state-of-the-art models, which can be categorized into four types: *Common models*: Multi-layer Perceptron (MLP), node2vec (n2v); *Un/self-supervised GNNs*: DGI, MVGRL, DIMP; *Semi-supervised GNNs*: GCN, GAT, GraphSAGE, JK-Net, GIN, GCNII, SGC, APPNP, PNA; *Enhancement techniques for semi-supervised GNNs*: CG^3, GRAND, GAM, GAUG, AdaEdge, CoCoS. A brief introduction of each model refer to **Appendix D.2**.

Implementations

We implement different GNN backbones for Violin in experiments. The learning rate is 0.01 for most variants. $\gamma = 0.4$ and $m = 2$ (the number of VO per node) for Ogbna while $\gamma = 0.6$ and $m = 1$ for others. α is searched in $[0.2, 1]$ by an interval of 0.2. We run each implementation ten times with ten random seeds. In each run, we record the test accuracy corresponding to the best validation loss and report the averaged test accuracy and standard deviation over ten runs. More implementation details of each variant can be found in **Appendix D.3**.

5.2 Experiments of Node Classifications

Quantitative Results

Since previous works mostly use GCN as the backbone, we mainly compare Violin's GCN variant with them for fairness. Table 3 shows the results on four datasets. We can observe that all Violin variants consistently outperform others or be competitive to the best on the board. Compared with unsupervised models, Violin can be trained in an end-to-end manner. This makes it more effective to learn from the supervised signals in the graph and yield task-oriented representations, which can be more discriminative in semi-supervised scenarios. The superiority of Violin over semi-supervised baselines is more significant. This is because Violin not only learn from the few labeled nodes but also all unlabeled but available data. More diverse knowledge and discriminative patterns can therefore be captured by Violin, which significantly improves the representation power of a GNN model.

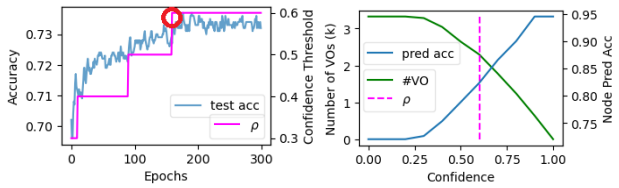

Figure 4: **Left**: The prediction accuracies and adaptive thresholds of Violin(GCN) on Citeseer during training. **Right**: The cumulative number of qualified nodes/ VOs and prediction accuracies at each confidence level, at the epoch marked by the red circle in the **Left**.

As an enhancement technique, Violin shows superiorities over other state-of-the-arts. First, Violin is a plug-and-play framework for common GNNs. Compared with other methods, Violin does not need any additional learnable parameters but only to fine-tune the backbone model, which can be more computationally effective and memory-efficient. Furthermore, for other competitors, such as GRAND and CG^3, elaborate handcrafted augmentations are required for generating qualified graph augmentations. In contrast, Violin automatically learns to generate the graph augmentation, i.e., the semantics-consistent graph. Such a data-driven scheme can be more adaptive to different scenarios, which results in learning more task-oriented representations.

Qualitative Analyses of Violin

To qualitatively investigate the effect of Violin, we monitor the value of ρ (the confidence threshold) during training. The left of Figure 4 shows an example, where ρ varies at different epochs. This is reasonable as the distribution of prediction confidence shifts as we train (refer to **Appendix E.4** for some examples). To match such dynamics, Violin adaptively adjusts ρ, which recalibrates itself to continually improve the model. We additionally inspect the status of the model at the epoch marked by the red circle, as shown in the right of Figure 4. Prediction accuracy rises in the intervals with higher confidence, but in the meanwhile, the number of qualified nodes (or VOs) drops. A large confidence threshold ensures that added VOs are almost correct but only few candidates fulfill, while a small threshold acts oppositely. However, the ρ of Violin is near the two curves' intersection. In other words, it adaptively makes a balance between the quantity and quality of VOs, which makes Violin effective to a GNN. More visualizations/ analyses refer to **Appendix E.5**.

5.3 Ablation Studies

Enhancement for Different GNN Backbones

To investigate the general enhancement performance of Violin, we extend it to more typical GNNs. Table 4 shows the experimental results. Obviously, almost all backbones are significantly enhanced and Violin still works for more advanced models (e.g., GCNII). It is worth mentioning that the original PNA model suffers from overfitting in Cora, Citeseer and Pubmed. However, with Violin, overfitting is well alleviated,

Backbone	Cora	Citeseer	Pubmed	Ogbna
Violin(GIN)	81.41(+3.47)	71.40(+4.53)	80.74(+2.91)	64.37(+1.18)
Violin(PNA)	81.41(+6.36)	62.50(+7.46)	79.58(+3.67)	OOM
Violin(JK-Net)	83.77(+3.42)	71.82(+4.53)	80.12(+1.76)	72.26(+0.07)
Violin(GAT)	84.33(+1.57)	72.08(+0.93)	79.12(+1.38)	71.41(−0.34)
Violin(SAGE)	84.38(+2.65)	73.07(+3.21)	78.99(+1.79)	72.11(+0.07)
Violin(APPNP)	84.41(+1.28)	72.64(+1.25)	**82.08(+1.78)**	71.51(+0.29)
Violin(GCN)	85.22(+2.70)	73.38(+2.36)	81.11(+2.36)	72.49(+0.50)
Violin(GCNII)	**85.63(+1.46)**	**74.19(+1.73)**	81.83(+1.98)	**73.18(+0.72)**

Table 4: Node classification accuracy (%) of Violin with different GNN backbones. (OOM: out of memory.)

Settings	Cora	Citeseer	Pubmed
GCN	82.52	71.02	79.16
Violin($\gamma = \alpha = 0$)	82.76(+0.24)	72.43(+1.41)	79.91(+0.75)
Violin($\gamma = 0$)	84.80(+2.28)	72.71(+1.69)	79.96(+0.80)
Violin($\alpha = 0$)	83.77(+1.25)	73.29(+2.27)	80.76(+1.60)
Violin($\rho = 0$)	84.73(+2.21)	73.07(+2.05)	80.18(+1.02)
Violin($\rho = 0.3$)	84.96(+2.44)	73.06(+2.04)	80.17(+1.01)
Violin($\rho = 0.6$)	84.72(+2.20)	73.32(+2.30)	80.42(+1.26)
Violin($\rho = 0.9$)	83.43(+0.91)	70.99(−0.03)	80.70(+1.54)
Violin (Adaptive ρ)	85.22(+2.70)	73.38(+2.36)	81.11(+2.36)

Table 5: Accuracies (%) of Violin(GCN) with different settings.

and the performance is improved by a large margin. This is because Violin learns not only from the few labeled nodes, but also from all available data, which encourages it to yield representations with better generalization ability. The great improvement also reflects that the learning capacity of the listed GNNs is under-exploited, while introducing Violin releases their potentials. In addition, we will show (in the following experiments) that the performance improvement can be more significant on Ogbna if only limited labels are available.

The Performance of Regularization Terms

We additionally evaluate the two regularization terms of Violin, i.e., $\mathcal{L}_{vo}$ and $\mathcal{L}_{con}$ in Equation 9. We train Violin(GCN) by removing the corresponding term but keep other settings the same. Results are shown in Table 5. Compared with the vanilla GCN, these two terms consistently improve the performance of Violin. $\mathcal{L}_{vo}$ keeps consistency between the two end-points of VOs, which induces Violin to mine consistent semantics within the graph. $\mathcal{L}_{con}$ forces the model to adapt to different graph augmentations, encouraging Violin to generate more robust node representations. These two regularization terms are complementary to each other, which jointly improves the performance of a model.

Quantitative Studies on the Confidence Threshold

We further investigate the effect of the confidence threshold ρ for Violin. Specifically, we train Violin(GCN) with a fixed ρ all the time. The results are shown in Table 5. Violin still works with most static thresholds. However, there does not exist a static threshold that can fit different datasets. In contrast, Violin with adaptive threshold dominates all. The adaptive mechanism continually monitors the confidence distributions and finds an appropriate threshold to fit the current training status when the confidence distribution shifts. This makes it easier to generalize to different scenarios and therefore learn discriminative representations for the task. More analyses on confidence thresholds refer to **Appendix E.6.3**.

Dataset	Ogbna		AmzP	CoCS
Train/Val/Test (%)	1/10/89	0.5/10/89.5	1.3/13.1/85.6	0.8/8.2/91
GCN	63.85	60.89	82.89	88.76
Violin(GCN)	66.82(+2.97)	64.88(+3.99)	87.77(+4.88)	91.82(+3.06)

Table 6: Prediction accuracies (%) with limited labels.

5.4 Evaluations with Limited Labels

We additionally investigate the performance of Violin(GCN) on Ogbna, AmzP and CoCS with limited labels. Specifically, on Ogbna, we randomly sample 10% of nodes for validation, 1% and 0.5% respectively for training, and the rest for testing. On AmzP and CoCS, we randomly sample 10 and 100 nodes of each class to form the training and validation set respectively, while the rest is used for testing.

Table 6 shows that Violin consistently boosts GCN across different low-label rate datasets. More importantly, the performance gain brought by Violin can be more significant when there are fewer labels. On Ogbna, Violin(GCN) trained on the $0.5/10/89.5$ split can even outperform the vanilla GCN trained with more labels (the $1/10/89$ split). This is because the introduction of VOs well improves the data utilization on unlabeled nodes. More general patterns can be captured from the graph and integrated with supervised signals to help classifications. Such results demonstrate that Violin is effective to learn from all available data under limited labels scenarios. Moreover, it can even achieve over 10% improvement on other backbones (e.g., GAT and GCNII). When under more extreme settings (e.g., 4 labels/ class for training), Violin can still outperform other state-of-the-arts methods. More details refer to **Appendix E.1** $\sim$ **E.3**.

5.5 Extended Experiments and Analyses

We conduct more experiments to investigate Violin's characteristics, including its 'sweet spot', robustness against noisy estimations, complexity, etc. **Appendix C** and **E** state details.

6 Conclusions

In this paper, we propose a generic framework, Violin, to enhance common GNNs for semi-supervised learning. By learning to add virtual overbridges (VOs) between two nodes, Violin can learn from unlabeled data and purposefully extends the receptive field towards regions informative to a target node. A GNN can therefore capture more discriminative patterns while sharing diverse knowledge with more nodes, which significantly improves the learning capacity. In addition, we propose an adaptive thresholding mechanism and integrate consistency regularization training to further enhance the model. The generated representations can therefore be more robust with better representation power. Violin can be readily extended to a wide range of GNNs with little computational overheads. Empirical evaluations show it robust to noisy VOs and adaptive to various of situations with limited labeled data. Extensive experiments on six datasets demonstrate its great effectiveness. As Violin is orthogonal to other enhancement methods, it is also feasible to integrate it with other state-of-the-arts to further improve GNN's representation power.

Acknowledgements

This research is supported in part by the Innovation and Technology Committee of HKSAR under the project#ITS/244/16, the CUHK MobiTeC R&D Fund and a grant from the graduate school of the Chinese University of Hong Kong.

References

[Chen *et al.*, 2020a] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[Chen *et al.*, 2020b] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

[Corso *et al.*, 2020] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, 2020.

[Feng *et al.*, 2020] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural network for semi-supervised learning on graphs. In *NeurIPS'20*, 2020.

[Gligorijević *et al.*, 2021] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):1–14, 2021.

[Hassani and Khasahmadi, 2020] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[Huang *et al.*, 2020] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Klicpera *et al.*, 2018] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[Klicpera *et al.*, 2019] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in Neural Information Processing Systems*, 32:13354–13366, 2019.

[Li *et al.*, 2017] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[Li *et al.*, 2020] Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1677–1688. IEEE, 2020.

[Rong *et al.*, 2019] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[Sun *et al.*, 2019] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2019.

[Sun *et al.*, 2020] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5892–5899, 2020.

[Thakoor *et al.*, 2021] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. *arXiv preprint arXiv:2102.06514*, 2021.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Veličković *et al.*, 2018] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[Wan *et al.*, 2020] Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. *arXiv preprint arXiv:2009.07111*, 2020.

[Wu *et al.*, 2019] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[Xie *et al.*, 2022] Siyue Xie, Wing Cheong Lau, et al. Cocos: Enhancing semi-supervised learning on graphs with unlabeled data via contrastive context sharing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4272–4280, 2022.

[Xu *et al.*, 2018a] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[Xu *et al.*, 2018b] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[Xu *et al.*, 2021] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[You *et al.*, 2021] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.

[Zhang *et al.*, 2018] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

[Zhao *et al.*, 2021] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11015–11023, 2021.

[Zhu *et al.*, 2021] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.