CORRUPTION-FREE SINGLE-VIEW SELF-SUPERVISED LEARNING ON GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised learning (SSL) for graphs is an essential problem since graph data are ubiquitous and data labeling is costly. We argue that existing SSL approaches for graphs have two limitations. First, they rely on corruption techniques such as node attribute perturbation and edge dropping to generate graph views for contrastive learning. These unnatural corruption techniques require extensive tuning efforts and provide marginal improvements. Second, the current approaches require the computation of multiple graph views, which is memory and computationally inefficient. These shortcomings of graph SSL call for a corruption-free single-view learning approach, but the strawman approach of using neighboring nodes as positive examples suffers two problems: it ignores the strength of connections between nodes implied by the graph structure on a macro level, and cannot deal with the high noise in real-world graphs. We propose CURSIVE, a corruption-free single-view graph SSL approach that overcomes these problems by leveraging graph diffusion to measure connection strength and denoise. With extensive experiments, we show that CURSIVE achieves up to 4.55% absolute improvement in ROC-AUC on graph SSL tasks over state-of-the-art approaches while being more memory efficient. Moreover, CURSIVE even outperforms supervised training on node classification tasks of ogbn-proteins dataset.

1 INTRODUCTION

Graph Neural Networks (GNN) (Welling & Kipf, 2016; Hamilton et al., 2017) are neural network architectures that extract meaningful and useful representations out of graph data. GNNs have shown great potential in a variety of fields including social networks (Fan et al., 2019; Chaudhary et al., 2019; Min et al., 2021), recommendation systems (Wu et al., 2020; 2019; Chang et al., 2021), and drug discovery (Jiang et al., 2021; Xiong et al., 2021; Jiang et al., 2021; Chen et al., 2018).

The Need for Self-Supervised Learning: Traditional supervised GNN training strategies require intensive data labeling, which is prohibitively expensive in some essential applications such as biochemistry (Xiong et al., 2019). As an alternative, Self-Supervised Learning (SSL) strategies do not rely on labels and have shown promising potential in graph learning. Prior SSL approaches such as DGI (Velickovic et al., 2019), GRACE (Zhu et al., 2020), BGRL (Thakoor et al., 2022) can learn meaningful representations that are useful in downstream tasks such as academic paper categorization, molecule classification, and product recommendation.

Problems of Existing Graph SSL Approaches: In this paper, we identify two problems in the current graph SSL approaches. First, prior SSL approaches for graphs rely on corruption techniques, which perturb node attributes or the adjacency matrix. The corruption techniques are inspired by data augmentation tricks from the computer vision (Shorten & Khoshgoftaar, 2019). However, unlike images, corrupted graphs may not maintain the original semantics at the node level or graph level. As a result, the encoder may not be able to learn meaningful representations because the learning goal is flawed. Second, existing graph SSL approaches need to compute multiple views of the graph, which increase the memory and computation complexity during training. This efficiency issue would be exacerbated when we train on large graphs with limited memory budget.

Given the limitation of current graph SSL approaches, it is natural to ask the following question.

Can we have a corruption-free single-view approach for graph SSL with promising performance?

In this paper, we answer this question positively by proposing CURSIVE, a corruption-free single-view graph SSL approach. In particular, we summarize our contributions as:

1. We propose a novel graph SSL approach by leveraging the diffusion matrix as the learning target for similarity between nodes from a single uncorrupted graph view. Without corruption, our proposed method leverages a natural learning objective to achieve significantly better accuracy with minimal tuning. Using only a single view, our approach is much more memory-efficient than previous methods and able to scale to large graphs that are impossible for multi-view methods.
2. We introduce an extension to our approach to easily scale CURSIVE to large-scale graphs by leveraging recent advances in efficient graph training. We scale SSL to large-scale graphs that are difficult to tune and time-consuming to train for the existing SSL methods.
3. Our extensive experiments demonstrate that CURSIVE achieves state-of-the-art accuracy on a variety of datasets. We highlight that our approach achieves 2.6% absolute improvement on PubMed, 4.55% absolute improvement on ogbn-proteins, and 3.04% absolute improvement on ogbn-products over the previous best.

The following sections are organzied as follows. We introduce the graph SSL problem and existing SSL methods' two major drawbacks in Section 2. The motivation behind our approach and the details of our proposed method are included in Section 3. We report the setup and results of our extensive experiments for evaluating our method in Section 4.

2 Graph Self-Supervised Learning

In this section, we introduce the graph self-supervised learning problem studied in this paper. Next, we identify the problems with two techniques of existing graph SSL methods.

2.1 Problem Formulation

Graph SSL aims to learn a GNN-based encoder that produces high-quality representations for graph data without using labels. We follow the standard problem setup of graph SSL (Velickovic et al., 2019; Zhu et al., 2020; Thakoor et al., 2022) to keep the training and evaluation procedures consistent with prior approaches. During the training stage, we have access to graph data $(\mathbf{X}, \mathbf{A})$ for training a GNN-based encoder $\mathcal{E}$, where $\mathbf{X} \in \mathbb{R}^{n \times k}$ is the node feature matrix and $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix. We denote each row of $\mathbf{X}$ as $\mathbf{x}_i$, which corresponds to a k-dimensional feature vector of node i, where $i \in [n]$. There should be an SSL objective to update the parameters of the encoder $\mathcal{E} : \mathbb{R}^{n \times k} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times d}$, which encodes the graph $(\mathbf{X}, \mathbf{A})$ into node representation matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$. Here each row of $\mathbf{Z}$, denoted as $\mathbf{z}_i$, corresponds to the d-dimensional representation of node i. We evaluate graph SSL methods by training and testing a linear classifier with the learned node representation matrix $\mathbf{Z}$ on downstream tasks.

2.2 Graph Corruption Techniques for Contrastive Learning

Prior competitive graph SSL methods rely on corrupting the input graph to generate positive and negative examples for learning. Graph corruption techniques perturb node attributes or the adjacency matrix to produce alternative graph views (Zhu et al., 2020). In this way, the GNN-based encoder (see Section 2.1) can learn to produce invariant representations. Popular graph corruption techniques include node feature masking (Zhu et al., 2020), node feature shuffling (Velickovic et al., 2019), node dropping (You et al., 2020), edge dropping (Zhu et al., 2020), and subgraphing (You et al., 2020). For example, CCA-SSG (Zhang et al., 2021) and BGRL (Thakoor et al., 2022) employ node feature masking and edge dropping to generate graph views and maximize the agreement between those views, DGI (Velickovic et al., 2019) uses node feature shuffling to produce negative examples, and GRACE (Zhu et al., 2020) uses node feature masking and edge removal for generating inter-view positives, inter-view negatives and intra-view negatives for contrastive learning. The graph corruption techniques are directly inspired by data augmentation methods from the computer vision domain, such as random erasing (Zhong et al., 2020) and cropping (Shorten & Khoshgoftaar, 2019).

However, corruption techniques for vision and graphs have a fundamental difference: corruptions of natural images preserve their underlying semantics, while the properties of a graph may alter significantly after minor corruptions. For example, dropping the edge to a hub node in a social network or citation graph may change the semantics of connected nodes significantly, and node and edge perturbations in molecular graphs can drastically change the properties of the molecule (Sun et al., 2021). Through extensive experiments, You et al. (2020) demonstrate that edge perturbations in graph SSL significantly degrade the model performance on molecular graphs. It is unclear which corruption techniques are applicable in different graphs, and finding a decent graph corruption requires significant trials and errors since many graphs are highly sensitive to corruption techniques and parameters. As a result, previous works (You et al., 2021; Thakoor et al., 2022; Zhang et al., 2021) resort to extensive grid search for the best combinations of corruption schemes, and show that different datasets require vastly different corruption parameters since the performance of learned models differ greatly with slight changes to corruption schemes and parameters.

2.3 Multi-view Representation Learning on Graphs

In addition to over-reliance on corruption techniques, prior graph SSL approaches compute multiple views of the same graph, which has significant memory and computational overhead. The computation of multiple views are required for previous methods since they mine positive/negative examples from them. For example, DGI (Velickovic et al., 2019) computes an additional view through shuffling node features to produce negative examples, LaGraph (Xie et al., 2022) computes two views and minimizes the distance between them, and BGRL Thakoor et al. (2022) computes four views for positive-only contrastive learning. This creates significant concerns related to computational efficiency and scalability. Modern hardware used for GNN training such as GPU has limited memory, and hence the computation of multiple views scale poorly to large graphs. Compared to supervised training which only computes a single view of the graph, prior self-supervised methods consume multiple times more memory and computation time. This is problematic in many real-world problems since common citation, co-purchasing, and social network graphs contain up to billions of nodes and edges (Hu et al., 2020). Although sub-sampling techniques can fit multiple views of the graph in a limited memory budget, they have been demonstrated to hurt performance significantly (Thakoor et al., 2022). As a result, it is ideal to have a graph SSL method that computes only a single view of the graph so that it can scale to larger graphs efficiently.

3 CURSIVE

In this section, we first motivate our proposed method by describing a strawman approach and identify its flaws in 3.1. Then we describe our proposed method by explaining our learning target in Section 3.2 and training strategy in Section 3.3. We then analyse the memory efficiency of our proposed approach and compare it against existing methods in Section 3.4. Finally, we describe an extension to our method for scaling to very large graphs in Section 3.5.

3.1 Our Motivation: A Corruption-free Single-view SSL on Graphs

The drawbacks of corruption techniques and multi-view approaches call for a corruption-free single-view SSL approach for graphs. However, this is nontrivial in practice.

3.1.1 A Strawman Approach

A strawman approach for corruption-free single-view SSL is to perform contrastive learning using neighboring nodes as positive instances and non-neighboring nodes as negative instances. But this method has two major problems: it only sees the local structure and fails to take into account the graph structure at a macro level, and it is impacted by noise in real-world datasets.

In this strawman approach, neighboring nodes are considered as positive examples and their mutual information is maximized. However, this is counterproductive because it ignores the rich information implied by the graph structure at a macro level. The strengths of connections between nodes vary greatly. For example, a graph may consists of a few densely connected node clusters with sparse edges across clusters. This is a common structure for many real-world graphs such as citation and social networks Fan et al. (2019). The graph structure on a macro level implies nodes are

more strongly linked to nodes within the same cluster, with weaker connection to nodes in other clusters. However, this information is not captured in the strawman approach, in which each edge is considered as equally strong. This hinders learning and discourages the encoder from understanding the global graph structure. As a result, the learned node representations are disentangled, shown in Figure 1. Furthermore, real-world graphs often include a large amount of noisy edges (Kang et al., 2019). The existence of noisy edges confuses the encoder and results in poorer representations.

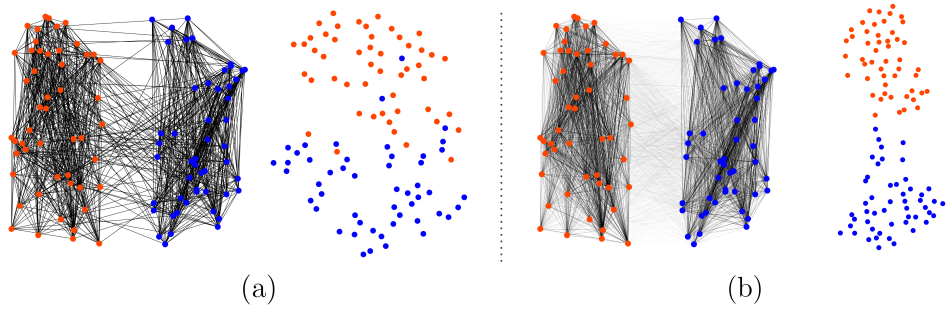

Figure 1: (a) Left: A graph of two densely connected clusters (orange and blue) with sparse edges across clusters. Right: t-SNE plot of the learned node representation using the strawman approach, in which clusters are not perfectly separated. (b) Left: The connection strengths measured by the diffusion matrix. Right: t-SNE plot of the learned node representations using CURSIVE, in which clusters are linearly separable.

3.1.2 Our Proposal

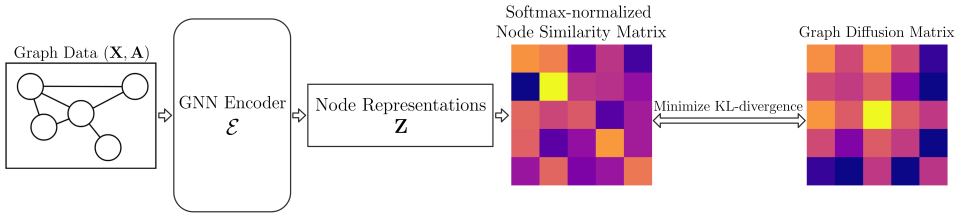

Figure 2: Our proposed training approach CURSIVE for graph SSL. During each training step, we feed-forward the uncorrupted graph a single time, and minimize the divergence between normalized node similarity and graph diffusion.

We propose to use the graph diffusion matrix as the learning target to resolve the two mentioned problems and incorporation the knowledge of graph structure into learned representations. Diffusion measures the flow of matter or information from one node to another through the global graph structure. It has direct connections to natural sciences (Vassilevich, 2003). For example, heat graph diffusion generalizes the kernel used for measuring the flow and distribution of heat to discrete graph structure (Chung, 2007). The continuous distribution of matter or information measured by the diffusion matrix can be seen as a measure of connection strength between nodes. This idea has been leveraged in web page rankings by using personalized PageRank as a diffusion matrix to measure the relative importance of web pages (Page et al., 1999). Thus, diffusion takes into account the global structure of the graph by emphasizing important connections while weakening unimportant ones. Moreover, the diffusion matrix has been shown to act as a denoising filter for graph structure to smooths out noisy edges (Gasteiger et al., 2019b). The diffusion matrix significantly boosts the signal-to-noise ratio of a graph.

Instead of maximizing mutual information between positive examples, we view the normalized similarity measure between a node and all other nodes as a distribution, and consider each row of the row-stochastic diffusion matrix as the target distribution, and minimize the divergence between the two distributions. This way, we do not use explicit positive and negative examples for learning, but we tune the relatedness of the learned node representations using the connection strength implied by the graph structure at a macro level.

Intuition In a graph, information in the neighbors around a node is important for the node's learned representation. Information should be spread from one node to other nodes across the graph in a natural process similar to how heat diffuses through a medium. With the diffusion matrix, we are able to measure the amount of information a source node should spread to other nodes. Since the diffusion matrix is symmetric, the amount of information spread from node u to node v is equal to the amount spread from v to u. We can use the similarity of two nodes' representations to measure the amount of their shared information. Therefore, if we enforce the similarity between nodes to follow the diffusion matrix, we can encourage the representation of each node to capture the ideal amount of information from nodes in the neighborhood around it.

3.2 Graph Diffusion Matrix as the Target

We adopt the definition of generalized graph diffusion proposed by Gasteiger et al. (2019b), which is defined as a convergent, infinite sum of weighted i-hop normalized adjacency matrices with self-edges, $\mathbf{T} = \sum_{i=0}^{\infty} \alpha_i \hat{\mathbf{A}}^i$. We use the symmetric normalization scheme proposed in Welling & Kipf (2016) for the adjacency matrix with self-edges, i.e., $\hat{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$, where $\mathbf{D}$ is the diagonal degree matrix. We consider two types of graph diffusion: heat diffusion (Chung, 2007) and personalized PageRank (PPR) Page et al. (1999). For the weighting coefficients, heat diffusion uses an exponential series $\alpha_i^{\text{heat}} = \frac{e^{-t}t^i}{i!}$, where t is the diffusion time (Chung, 2007), and PPR diffusion uses a geometric series $\alpha_i^{\text{PPR}} = \beta(1-\beta)^i$, where β is the teleport probability (Page et al., 1999). Heat diffusion is a concept commonly applied in natural sciences to measure the distribution of heat or diffusive matter (Vassilevich, 2003), and Chung (2007) generalized the heat kernel to discrete graph structures. PPR diffusion is interpreted as a probability matrix in which the entry u, v is the probability of a random walk starting at node u with terminating probability β at each step eventually terminating at node v, and it is used in search engines to measure the personalized importance of a web page (Page et al., 1999). Both forms of diffusion have closed-form solutions, or they can be approximated by computing the sum of only the first few terms in $\sum_{i=0}^{\infty} \alpha_i \hat{\mathbf{A}}^i$.

Following the definition above, the diffusion matrix $\mathbf{T}$ is a row-stochastic matrix, and hence each row of $\mathbf{T}$ can be viewed as a distribution. Specifically, we define the diffusion distribution of node u as $\mathbb{T}_u(v) = \mathbf{T}_{u,v}, v \in [n]$, where $\mathbf{T}_{u,v}$ is the value in the uth row and vth column of the diffusion matrix. We will use the diffusion distribution of nodes as the target of our learning objective.

Node Similarity To measure the similarity between the learned representations of two nodes u, v, we use dot product which is a symmetric similarity score, $\text{sim}(\mathbf{z}_u, \mathbf{z}_v) = \sum_i \mathbf{z}_{ui}\mathbf{z}_{vi}$. We apply softmax normalization to normalize the similarity scores between node u and all other nodes into a distribution. Specifically, the similarity distribution of node u is

$$\mathbb{S}_u(v) = \frac{\exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_v))}{\sum_{i=1}^{n} \exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_i))}, \text{where } v \in [n].$$

3.3 Training Strategy

Loss Function Our proposed objective minimizes the divergence between the diffusion distribution and the similarity distribution of node u, for all $u \in [n]$. Therefore, we construct our loss function as the mean Kullback–Leibler divergence (Kullback & Leibler, 1951) between the diffusion distribution and the similarity distribution, $\frac{1}{n}\sum_{u=1}^{n} D_{\text{KL}}(\mathbb{T}_u \parallel \mathbb{S}_u) = \frac{1}{n}\sum_{u=1}^{n}(\sum_{v=1}^{n} \mathbb{T}_u(v)\log \mathbb{T}_u(v) - \sum_{v=1}^{n} \mathbb{T}_u(v)\log \mathbb{S}_u(v))$. Since the entropy of the diffusion distribution of a given graph is fixed (i.e. $\sum_{v=1}^{n} \mathbb{T}_u(v)\log \mathbb{T}_u(v)$ is a constant value), we omit it and equivalently minimize the following loss function. An overview of CURSIVE's training strategy is shown in Figure 2.

$$\mathcal{L} = -\frac{1}{n}\sum_{u=1}^{n}\sum_{v=1}^{n} \mathbf{T}_{u,v} \log \frac{\exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_v))}{\sum_{i=1}^{n} \exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_i))}$$

In practice, mini-batch gradient descent help convergence and reduce memory usage (Ruder, 2016). Therefore, in each training step, we sample a batch of node indices $B = \{b_1, \cdots, b_{|B|}\}$ from

$\{1, \cdots, n\}$ without replacements, and minimize the following batched loss function.

$$\mathcal{L}_{\text{batch}} = -\frac{1}{|B|} \sum_{u \in B} \sum_{v=1}^{n} \mathbf{T}_{u,v} \log \frac{\exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_v))}{\sum_{i=1}^{n} \exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_i))}$$

With batched execution, only $|B|$ rows of the diffusion matrix needs to be loaded into GPU memory, which reduces the memory complexity of the loss computation from $O(n^2)$ to $O(|B|n)$. Since the batched loss function and the original loss function are equal in expectation, i.e. $E(\mathcal{L}_{\text{batch}}) = \mathcal{L}$, the quality of the learned model does not degrade with batched execution. Empirically, we found batch sizes of 1024 and 2048 work well. Pseudo-code for CURSIVE is given in Algorithm 1.

Algorithm 1 Our proposed method CURSIVE for graph SSL

Input: node features $\mathbf{X} \in \mathbb{R}^{n \times k}$, adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, GNN encoder $\mathcal{E} : \mathbb{R}^{n \times k} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times d}$, number of epochs n_{epochs}
Output: trained GNN encoder $\mathcal{E} : \mathbb{R}^{n \times k} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times d}$
Compute the diffusion matrix $\mathbf{T}$ through the closed-form solution or approximation
for $e = 1 \cdots n_{epochs}$ **do**
  Let $N = \{1, \cdots, n\}$
  while $|N| > 0$ **do**
    Sample a batch of node indices $B = \{b_1, \cdots, b_{|B|}\}$ from N without replacements
    Let $\mathbf{Z}$ be the output of the encoder $\mathcal{E}(\mathbf{A}, \mathbf{X})$
    Compute $\mathcal{L} = -\frac{1}{|B|} \sum_{u \in B} \sum_{v=1}^{n} \mathbf{T}_{u,v} \log \frac{\exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_v))}{\sum_{i=1}^{n} \exp(\text{sim}(\mathbf{z}_u, \mathbf{z}_i))}$ and perform back-propagation to update the parameters of $\mathcal{E}$
  end
end
return $\mathcal{E}$

3.4 Memory Analysis

Without the reliance on multiple graph views or extra MLP layers, our approach has clear advantage in memory efficiency over prior approaches. Table 1 presents the memory complexity and the empirical GPU memory usage of the most competitive graph SSL methods on ogbn-arxiv and ogbn-proteins datasets (Hu et al., 2020). Each forward pass/back-propagation consumes $O(n + m)$ memory, where n is the number of nodes and m is the number of edges in the graph. We let C^{fw} be the constant factor for each forward pass and C^{bw} be the constant factor for each back-propagation. We consider the most memory-efficient graph SSL methods GRACE (Zhu et al., 2020), BGRL (Thakoor et al., 2022), LaGraph (Xie et al., 2022), and CCA-SSG (Zhang et al., 2021), all of which compute two or more graph views at each training step. GRACE uses intra-view and inter-view negative examples, and hence computing its loss function consumes $O(n^2)$ memory. BGRL does not use negative instances in its loss function to avoid quadratic blowup, but it computes 4 graph views in total, 2 by the online encoder and 2 by the target encoder. LaGraph and CCA-SSG both compute 2 graph views and maximize the invariance between views, and LaGraph uses an additional MLP component as the decoder. Our method computes a single graph view, with memory efficiency on par with supervised training theoretically and empirically.

We employ the same encoder (3-layer GCN (Welling & Kipf, 2016)) for all approaches to ensure a fair comparison of memory usage. On ogbn-arxiv, the memory efficiency of our method is on par with supervised training, while other methods consume $2\times$ or more memory. On ogbn-proteins, supervised training consumes more than half of the GPU memory, which makes multi-view training impossible. Therefore, only our SSL method is able to train on ogbn-proteins without running out of memory.

3.5 Scaling to Large Graphs

Many real-world graphs are so large that even memory-efficient supervised training has to resort sub-sampling techniques such as neighbor-sampling. For CURSIVE, neighbor sampling techniques do not work since the traditional neighbor sampling technique (Hamilton et al., 2017) does not preserve graph structure, so the diffusion matrix for the set of sampled nodes is not well defined. To

Table 1: Memory complexity and empirical GPU memory usage of competitive graph SSL approaches on ogbn-arxiv and ogbn-proteins datasets. "OOM" means out of memory on a GPU with 32 GB of memory.

Method	Memory Complexity	GPU Memory Usage	
		ogbn-arxiv	ogbn-proteins
Supervised GCN	$C^{\text{fw}}_{\text{GCN}}(n+m) + C^{\text{bw}}_{\text{GCN}}(n+m) + C_{\text{Supervised}} \cdot n$	6.9G	17.2G
GRACE	$2C^{\text{fw}}_{\text{GNN}}(n+m) + C^{\text{bw}}_{\text{GNN}}(n+m) + 2C^{\text{fw}}_{\text{MLP}} \cdot n + C^{\text{bw}}_{\text{MLP}} \cdot n + C_{\text{GRACE}} \cdot n^2$	OOM	OOM
BGRL	$4C^{\text{fw}}_{\text{GNN}}(n+m) + C^{\text{bw}}_{\text{GNN}}(n+m) + 2C^{\text{fw}}_{\text{MLP}} \cdot n + C^{\text{bw}}_{\text{MLP}} \cdot n + C_{\text{BGRL}} \cdot n$	25.4G	OOM
LaGraph	$2C^{\text{fw}}_{\text{GNN}}(n+m) + C^{\text{fw}}_{\text{GNN}}(n+m) + C^{\text{bw}}_{\text{MLP}} \cdot n + C^{\text{bw}}_{\text{MLP}} \cdot n + C_{\text{LaGraph}} \cdot n$	16.9G	OOM
CCA-SSG	$2C^{\text{fw}}_{\text{GNN}}(n+m) + C^{\text{bw}}_{\text{GNN}}(n+m) + C_{\text{CCA-SSG}} \cdot n$	15.6G	OOM
CURSIVE (ours)	$C^{\text{fw}}_{\text{GNN}}(n+m) + C^{\text{bw}}_{\text{GNN}}(n+m) + C_{\text{CURSIVE}} \cdot b \cdot n$	7.8G	18.3G

scale to very large graphs, we propose a natural extension to our method by leveraging recent advances in efficient GNN training. Cluster-GCN (Chiang et al., 2019) proposes to train on subgraphs partitioned with METIS (Karypis & Kumar, 1997) to avoid the exponential neighborhood expansion problem. During training, a random batch of subgraphs are merged to form the input to the GNN. The diffusion matrix is well-defined on the subgraphs, and we are able to leverage Cluster-GCN to scale CURSIVE to very large graphs. We evaluate the effectiveness of CURSIVE scaled to large graphs in Section 4.3.

4 Experiments

In this section, we evaluate the performance of CURSIVE and compare it against the most competitive graph SSL methods on a variety of node classification datasets. We first introduce the setup, the datasets, and the baselines used for the evaluation in Section 4.1. Then we present the evaluation results on small to medium-scale datasets in Section 4.2 and results on large-scale datasets in Sectin 4.3. Finally, we present the results of the ablation study in Section A.3 in the Appendix.

4.1 Settings

Evaluation Setup We take a untrained GNN encoder with randomly initialized parameters, and train it using CURSIVE and baseline methods on the graph data $(\mathbf{X}, \mathbf{A})$ without labels until convergence. Then we freeze the parameters of the GNN and and use it to encode the nodes into learned node representations $\mathbf{Z}$. We then train a linear classifier (logistic regression classifier) on the labelled training set with $\mathbf{Z}$ as input, and report the evaluation metrics on the unseen test set. Our evaluation setup is identical to previous works (Velickovic et al., 2019; Thakoor et al., 2022) to keep the evaluation fair and consistent. Details about the testbed for performing the evaluation is given in Section A.2 in the Appendix, and the hyper-parameters and training details are presented in Table 6 in the Appendix.

Datasets Our baselines are evaluated on 6 datasets, including 3 small scale datasets (Cora, Citeser, PubMed (Sen et al., 2008)), 1 medium scale dataset (ogbn-arxiv (Hu et al., 2020)) and 2 large scale dataset (ogbn-proteins and ogbn-products (Hu et al., 2020)). The graph statistics are summarized in Table 4 in the Appendix.

Baselines We perform a thorough comparison of CURSIVE against the current most competitive graph SSL methods: *1*) DGI (Velickovic et al., 2019) proposes to learn node representations by maximizing the mutual information between node representations and the global representation through contrasting representations of a corrupted graph. *2*) GATE (Salehi & Davulcu, 2019) reconstructs the input graph with an auto-encoder architecture that uses self-attention. *3*) GRACE (Zhu et al., 2020) performs contrastive learning on positive and negative examples from two different corrupted graph views. *4*) BGRL (Thakoor et al., 2022) learns contrastively from positive examples only by leveraging bootstrapping. *5*) LaGraph (Xie et al., 2022) learns through a reconstruction loss and an invariance loss between the representations of the original graph and a corrupted graph. *6*) GraphMAE (Hou et al., 2022) learns through reconstructing node features using two GNNs as encoder and decoder. *7*) InfoGCL (Xu et al., 2021) maximizes the agreement between the learned representations of two corrupted graph views encoded by a GNN and MLP. *8*) CCA-SSG (Zhang et al., 2021) maximizes the agreement between two corrupted graph views using a loss function inspired

by Canonical Correlation Analysis. We also include the most common supervised model baselines for reference, which are trained with the training set as supervision. *1*) MLP (Hu et al., 2020) is a multi-layer perceptron network with only the node features as input. *2*) GCN (Welling & Kipf, 2016) propagates node information through convolutional layers. *3*) GAT (Veličković et al., 2018) leverages self-attention to adaptively aggregate node information. *4*) GraphSage (Hamilton et al., 2017) aggregates node feature information to generalize to unseen data.

4.2 Evaluation on Small and Medium-scale Graphs

Table 2 presents the accuracy of CURSIVE and baselines on Cora, Citeseer, Pubmed and ogbn-arxiv. CURSIVE achieves state-of-the-art performance on all 4 datasets, and improves the previous best by 2.6% on PubMed, 1.1% on CiteSeer, 0.2% on Cora and 0.33% on ogbn-arxiv. Our method significantly exceeds the accuracy of supervised training on Cora, CiteSeer, PubMed, showing its potential in eliminating the reliance on labels in graph learning. On ogbn-arxiv, our method achieves accuracy competitive with the best supervised model GAT (within 0.02% difference) while using a simpler model (GCN).

Table 2: Performance of self-supervised learning methods in terms of classification accuracy (along with standard deviations). The results of baselines are taken from official papers. "OOM" means out of memory on a GPU with 32 GB of memory.

Learning Paradigm	Method	Dataset			
		Cora	Citeseer	PubMed	ogbn-arxiv
Supervised	MLP	55.1	46.5	71.4	55.50
	GCN	81.5	70.3	79.0	71.74 ± 0.29
	GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	72.10 ± 0.13
Self-supervised	DGI	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6	70.34 ± 0.16
	GATE	83.2 ± 0.6	71.8 ± 0.8	80.9 ± 0.3	OOM
	GRACE	81.9 ± 0.4	71.2 ± 0.5	80.6 ± 0.4	71.51 ± 0.11
	BGRL	82.7 ± 0.5	71.1 ± 0.8	79.6 ± 0.5	71.64 ± 0.12
	LaGraph	84.1 ± 0.3	73.0 ± 0.4	80.9 ± 0.3	71.71 ± 0.21
	GraphMAE	$\underline{84.2 \pm 0.4}$	73.4 ± 0.4	81.1 ± 0.4	$\underline{71.75 \pm 0.17}$
	InfoGCL	83.5 ± 0.3	$\underline{73.5 \pm 0.4}$	79.1 ± 0.2	OOM
	CCA-SSG	84.0 ± 0.4	73.1 ± 0.3	$\underline{81.2 \pm 0.3}$	71.24 ± 0.20
	CURSIVE (ours)	$\mathbf{84.4 \pm 0.1}$	$\mathbf{74.6 \pm 0.1}$	$\mathbf{83.8 \pm 0.1}$	$\mathbf{72.08 \pm 0.12}$

Table 3: AUC-ROC on ogbn-proteins and accuracy on ogbn-products of the best graph SSL methods. We consider GCN and GraphSage as the backbone model for each method. †: sub-sampling is used since multiple views of ogbn-proteins do not fit into GPU memory.

Learning Paradigm	Method	ogbn-proteins	ogbn-products
Supervised	GCN	72.51 ± 0.01	75.64 ± 0.01
	GraphSage	77.68 ± 0.01	78.29 ± 0.01
Self-supervised	InfoGCL	OOM	OOM
	GraphMAE	$62.52 \pm 0.69^\dagger$	72.88 ± 0.37
	GRACE	$68.40 \pm 0.59^\dagger$	71.55 ± 0.88
	LaGraph	$71.86 \pm 0.28^\dagger$	73.23 ± 0.25
	BGRL	$\underline{73.25 \pm 0.79^\dagger}$	72.86 ± 0.64
	CCA-SSG	$73.08 \pm 0.37^\dagger$	$\underline{73.46 \pm 0.26}$
	CURSIVE (ours)	$\mathbf{77.80 \pm 0.21}$ $(+4.55)$	$\mathbf{76.50 \pm 0.18}$ $(+3.04)$

4.3 Evaluation on Large-scale Graphs

We evaluate CURSIVE and baselines on ogbn-proteins and ogbn-products, which are two challenging large-scale node classification datasets. Only CURSIVE is able to train on ogbn-proteins using a single GPU without sub-sampling, other methods require sub-sampling to fit into 32 GB

of GPU memory since they rely on multiple graph views (see Section 3.4). We leverage the sub-sampling techniques described by Hamilton et al. (2017); Thakoor et al. (2022) to scale the baselines. CURSIVE achieves 4.55% better AUC-ROC than the current best SSL method. More importantly, CURSIVE beats supervised training: 5.29% and 0.12% better AUC-ROC than supervised GCN and GraphSage. As a biological graph, ogbn-proteins is sensitive to graph corruptions. Our method achieves the best accuracy by avoiding corruptions and training on the full graph, which maximally preserves semantics of the original graph.

The graph of ogbn-products is so large that even supervised training has to resort to sub-sampling. Therefore, we leverage Cluster-GCN (Chiang et al., 2019) to efficiently scale CURSIVE by partitioning the graph into 100 clusters. Our method exceeds the previous best SSL method by 3.04%. Furthermore, our method beats supervised training by 0.86% when using the same GCN architecture, suggesting our method has potential of eliminating the need for costly labels in graph learning.

5 Related Works

Self-supervised Learning for Graphs The success of self-supervised contrastive learning in computer vision (Oord et al., 2018; Hjelm et al., 2018; Grill et al., 2020) inspired the development of contrastive learning methods for graph SSL based on mutual information maximization. For example, DGI (Velickovic et al., 2019) maximizes mutual information between local patch representations and global graph representation by contrasting with negative examples from shuffled node features. GRACE (Zhu et al., 2020) maximizes the mutual information between node representations of two corrupted graph views by contrasting with intra- and inter-view negatives. BGRL (Thakoor et al., 2022) leverages BYOL (Grill et al., 2020) to perform contrastive learning without negative examples. InfoGCL (Xu et al., 2021) proposes a contrastive framework to maintain task-relevant information at different levels and minimize the information loss during graph representation learning. MVGRL (Hassani & Khasahmadi, 2020) uses graph diffusion to produce an alternative graph view and maximize the mutual information between the local representation of one view and the global representation of the other view. Although MVGRL also utilizes graph diffusion, it is very different from our work since it uses diffusion to replace the graph structure, and MVGRL is a multi-view multi-scale contrastive method. Graph SSL methods based on the reconstruction objective have also been proposed in the past. For instance, GATE (Salehi & Davulcu, 2019) uses stacked self-attention-based encoder/decoder architecture to reconstruct node features and graph structure. GraphMAE (Hou et al., 2022) proposes to focus on feature reconstruction using a graph autoencoder. Recently, predictive graph SSL methods have also been proposed. LaGraph (Xie et al., 2022) proposes to learn through predicting unobserved latent graphs. CCA-SSG (Zhang et al., 2021) leverages a feature prediction objective inspired by canonical correlation analysis.

Graph Diffusion Graph diffusion has been used extensively in search engines for ranking web pages (Page et al., 1999; Chung, 2007). Diffusion has also been applied to graph learning in the past, but previous works focus on changing the architecture of GNNs using diffusion. For example, Gasteiger et al. (2019b) proposes graph diffusion convolution (GDC) that replaces the normalized adjacency matrix with diffusion matrix in GCN to expand receptive field. Personalized propagation of neural predictions (PPNP) (Gasteiger et al., 2019a) is an improved propagation scheme based on personalized PageRank (PPR). PPRGo (Bojchevski et al., 2020) leverages an efficient approximation to PPR to scale PPNP to large-scale graphs without compromising on accuracy.

6 Conclusion

We propose CURSIVE, a corruption-free single-view approach for graph SSL. By avoiding corruption techniques employed by prior SSL method, we achieve a natural SSL objective to attain significantly better accuracy on a variety of datasets with minimal tuning efforts. Without the computation of multiple views, our method is more memory-efficient and scalable than prior approaches and able to scale to large graphs that are difficult for existing methods. Through extensive experiments, we demonstrate the advantage of CURSIVE over existing SSL methods with significant improvements in accuracy on popular graph benchmarks. Furthermore, our SSL approach is able to surpass or be competitive with the accuracy of fully supervised training on large-scale datasets, which is a crucial step towards eliminating the need for costly labels in graph learning.

REFERENCES

Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2020. ACM.

Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 378–387, 2021.

Anshika Chaudhary, Himangi Mittal, and Anuja Arora. Anomaly detection using graph neural networks. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 346–350. IEEE, 2019.

Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 257–266, 2019.

Fan Chung. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences*, 104(50):19735–19740, 2007. doi: 10.1073/pnas.0708838104. URL https://www.pnas.org/doi/abs/10.1073/pnas.0708838104.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019b.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

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

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proceedings of International Conference on Machine Learning*, pp. 3451–3461. 2020.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 594–604, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539321. URL https://doi.org/10.1145/3534678.3539321.

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

Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23, 2021.

Zhao Kang, Haiqi Pan, Steven CH Hoi, and Zenglin Xu. Robust graph learning from noisy data. *IEEE transactions on cybernetics*, 50(5):1833–1843, 2019.

George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. Stgsn — a spatial–temporal graph neural network framework for time-evolving social networks. *Knowledge-Based Systems*, 214:106746, 2021. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2021.106746. URL https://www.sciencedirect.com/science/article/pii/S0950705121000095.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

Amin Salehi and Hasan Davulcu. Graph attention auto-encoders. *arXiv preprint arXiv:1905.10715*, 2019.

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

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. Mocl: data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3585–3594, 2021.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0UXT6PpRpW.

Dmitri V Vassilevich. Heat kernel expansion: user's manual. *Physics reports*, 388(5-6):279–360, 2003.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 346–353, 2019.

Yaochen Xie, Zhao Xu, and Shuiwang Ji. Self-supervised representation learning via latent graph prediction, 2022. URL https://openreview.net/forum?id=Da3ZcbjRWy.

Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019.

Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 30414–30425. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/ff1e68e74c6b16a1a7b5d958b95e120c-Paper.pdf.

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.

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

Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S. Yu. From canonical correlation analysis to self-supervised graph neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=X3TdREzbZN.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 13001–13008, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Contrastive Representation Learning. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020. URL http://arxiv.org/abs/2006.04131.

A APPENDIX

A.1 STATISTICS OF DATASETS

Table 4 presents the statistics of the 6 graph datasets used in our evaluations.

Table 4: Statistics of the graphs used in the experiments.

Dataset	Cora	Citeseer	PubMed	ogbn-arxiv	ogbn-proteins	ogbn-products
# of Nodes	2,708	3,327	19,717	169,343	132,534	2,449,029
# of Edges	5,429	4,732	44,338	1,166,243	39,561,252	61,859,140
Metric	Accuracy	Accuracy	Accuracy	Accuracy	ROC-AUC	Accuracy
# of Classes	7	6	3	40	112 (binary classification)	47

A.2 TESTBED

We implement our proposed method with the Deep Graph Library (Wang et al., 2019). Our experiments are conducted on a machine with 1 NVIDIA Tesla V100 32GB GPU, 2 24-core/48-thread Intel Xeon Gold 5220R CPUs, and 1.5TB of RAM.

A.3 ABLATION STUDY

We study the sensitivity of our method to hyper-parameter changes. A robust SSL method should not be sensitive to hyper-parameters. This has been a weakness of prior SSL methods, which require vastly different corruption parameters for different datasets (You et al., 2021; Thakoor et al., 2022; Zhang et al., 2021). We vary the hyper-parameters in computing the diffusion matrix for both types of diffusion matrix (diffusion time t for heat kernel and teleport probability β for PPR) and present the test accuracy on Cora, Citeseer and PubMed in Table 5. Our method is not sensitive to hyper-parameters of the diffusion target, since the accuracy drops at most 1.0% from the best accuracy for all diffusion parameters and diffusion types considered. Therefore, our method is more robust than previous SSL approaches which are sensitive to hyper-parameter changes.

Table 5: Evaluate performance of CURSIVE by varying the diffusion hyper-parameters.

Hyperparameter	Heat (t)					PPR (β)				
	3	4	5	6	7	0.02	0.04	0.06	0.08	0.10
Cora	83.9	84.1	84.0	84.0	**84.4**	84.1	84.0	84.3	84.1	84.0
Citeseer	73.3	74.1	74.3	74.1	74.1	74.0	**74.6**	74.2	74.0	74.2
PubMed	82.8	83.0	83.6	83.1	82.7	83.6	83.2	**83.8**	82.8	83.4

Table 6: Training details and hyper-parameters for CURSIVE on all datasets

	Cora	Citeseer	PubMed	ogbn-arxiv	ognb-proteins	ogbn-products
Architecture	GCN	GCN	GCN	GCN	GCN	GCN
Depth	1	1	2	3	3	3
Hidden Size	512	512	32	1024	1024	1024
Activation	Leaky ReLU	tanh	tanh	Leaky ReLU	Leaky ReLU	tanh
Learning Rate	5e-3	5e-4	1e-2	1e-4	1e-4	1e-4
Diffusion Type	Heat	PPR	PPR	PPR	PPR	PPR
Epochs	100	100	100	100	200	10
Optimizer	AdamW (Loshchilov & Hutter, 2019)					