ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240779

Every Node Counts: Improving the Training of Graph Neural Networks on Node Classification

Moshe Eliasof[a,*], Eldad Haber[b,] and Eran Treister[c,***]**

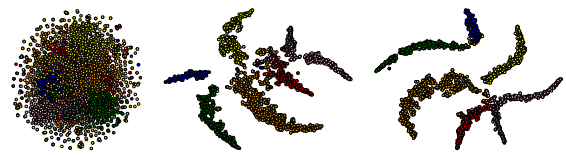
[a]Department of Applied Mathematics and Theoretical Physics, University of Cambridge
[b]Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia
[b]Department of Computer Science, Ben-Gurion University of the Negev

Abstract. Graph Neural Networks (GNNs) are prominent in handling sparse and unstructured data efficiently and effectively. Specifically, GNNs were shown to be highly effective for node classification tasks, where labelled information is available for only a fraction of the nodes. Typically, the optimization process, through the objective function, considers only labelled nodes while ignoring the rest. In this paper, we propose novel objective terms for the training of GNNs for node classification, aiming to exploit all the available data and improve accuracy. Our first term seeks to maximize the mutual information between node and label features, considering both labelled and unlabelled nodes in the optimization process. Our second term promotes anisotropic smoothness in the prediction maps. Lastly, we propose a cross-validating gradients approach to enhance the learning from labelled data. Our proposed objectives are general and can be applied to various GNNs, and require no architectural modifications. Extensive experiments demonstrate our approach using popular GNNs like Graph Convolutional Networks (e.g., GCN and GCNII), and Graph Attention Networks (e.g., GAT), reading a consistent and significant accuracy improvement on 10 real-world node classification datasets.

1 Introduction

[1] The field of Graph Neural Networks (GNNs) has gained large popularity in recent years [24, 39] in a variety of fields and applications such as computer graphics and vision [46], Bioinformatics [22], node classification [24, 39, 6] and others. In the context of node classification, many methods [24, 39, 6, 58] and others, consider the semi-supervised setting, where only a small part of the nodes are labelled, e.g., 20 labelled nodes per class in the Cora dataset, translating to 5% labelled nodes. Others consider the fully-supervised setting, where typically [30] 48% of the nodes are labelled. The recipes of the various methods share a common factor, which is the training procedure. While the aforementioned methods propose novel architectures, aggregation schemes or network dynamics, they yet require the minimization of the cross-entropy loss of *labelled* node predictions. This gives rise to the research question – it is beneficial to also consider the unlabelled information and specifically the predictions of unlabelled nodes in the training procedure, and how?

* Email: me532@cam.ac.uk.
** Email: ehaber@eoas.ubc.ca
*** Email: erant@cs.bgu.ac.il
[1] Our Appendix can be found in [12].



(a) (b) (c)

Figure 1: t-SNE embeddings of (a) Cora input features, (b) GCN predictions, (c) ENC-GCN predictions. Our ENC-GCN improves the node classification accuracy by 4.3% compared to GCN. Zoom in for a better view.

This question is often treated by requiring the consistency of the predicted node classification with respect to adversarial perturbations of the data [14], Laplacian-based regularization [51] of the node predictions, label entropy minimization [37, 2] and label propagation methods [43, 10]. Most of the aforementioned methods rely on designing novel objective functions to achieve the desired consistency and improved performance, highlighting that the research front of designing improved objective functions is an active field of research. In this paper, we focus on proposing additional objective terms to improve the performance of GNNs on node classification tasks. While the considered methods show significant improvement over the baseline, we identify that there are limitations that can be further relieved. For instance, demanding smoothness according to the Laplacian may be useful in the case of homophilic (following the definition from [30]) datasets like Cora, Citeseer and Pubmed. However, for datasets like Cornell, Texas and Wisconsin, that have a low homophily score (i.e., heterophilic datasets), such a demand can result in degraded accuracy, as we show in Section 5.3. An adversarial perturbation of the data is highly dependant on the perturbation policy, and requires additional computational costs due to the comparison of two or more forward computations of the network. Simple label entropy minimization can gravitate the network towards predicting a dominant class in the labelled data, which may not reflect the real distribution of the data.

In this paper, we propose to incorporate both labelled and unlabelled nodes in the objective functions through the perspectives of Mutual Information (MI) maximization, Total Variation (TV) regularization, as well as a Cross-Validating Gradients (CVG) approach that improves training from labelled nodes. We find that augmenting the standard cross-entropy objective function with unlabelled nodes information and enhanced labelled node information leads to a consistent improvement across all considered datasets and experiments,

showing that the information from *every node counts* to improve accuracy. We therefore call our method ENC. An example of the obtained node classification of our ENC approach compared to a baseline GCN is given in Figure 1. Our contributions are as follows:

- We propose a Mutual Information based objective function that aims to maximize prediction information between all nodes and classes for semi-supervised tasks.
- A Total Variation guided objective is proposed to promote node class predictions that adhere to the natural boundaries of the graph signal. Our theoretical understanding and experimental results show the efficacy of this loss compared to alternative smoothness terms.
- A novel Cross-Validating Gradients approach is introduced as a stochastic measure to improve the gradient direction from labelled nodes.
- Through a series of extensive experiments, we demonstrate the added value of our method, reading a consistent improvement on all datasets and baselines, that is in line with current state-of-the-art methods.

2 Related Work

2.1 Graph Neural Networks

Graph Neural Networks were introduced by [34], and became popular in recent years, with the rise of the the Message-Passing Neural Network (MPNN) mechanism by [16], where each node aggregates features (messages) from its neighbours, according to some policy. Most existing GNNs can be thought of MPNNs. For instance, GCN [24] and ChebNet [8] polynomials of the graph Laplacian to parameterize the convolution operator. Attention-based methods like GAT [39] and SuperGAT [23] learn a non-negative score of the graph edges to perform local propagation of node features. Computer Vision oriented methods like DGCNN [46] constructs a k-nearest-neighbours graph from point-clouds and dynamically updates it. A shared quality of the aforementioned methods is the training scheme. While all methods focus on minimizing a problem designated loss (e.g., the cross-entropy loss function for node or graph classification) with respect some labelled data, there is no consideration of the unlabelled data. In this paper we focus on the incorporation of unlabelled data to improve the training of GNNs on the node-classification task.

2.2 Improved training of Graph Neural Networks

The study of improved training of neural networks, and in particular of GNNs is concerned with creating different training policies and losses. Perhaps the most basic and common remedy for training on the typically small datasets like Cora, Citeseer and Pubmed is the incorporation of Dropout after every GNN layer, which has become a standard practice [24, 6, 58]. Other important tools that improve training are realized by randomly alternating the data rather than the neural units of the GNN. For instance, [31] suggest DropEdge, a method that randomly drops graph edges, and [9] propose DropNode – a method that randomly removes graph nodes. Other methods like PairNorm [56] propose adding node features normalization, which also helps to alleviate the over-smoothing phenomenon in GNNs discussed in [5]. Another approach is the Mixup [54] technique that enriches the learning data, and has shown success in image classification tasks. Following that, works like GraphMix [41] proposed an interpolation-based regularization method by parameter sharing of GNNs and point-wise convolution. The methods above were shown to improve the training in GNNs. However, they do not consider the information of the unlabelled nodes during the training.

Recent methods that consider both labelled and unlabelled data in the context of improved GNN training include InfoGraph [37] that learns a discriminative network for graph classification tasks. For graph classification, [53] suggest to utilize unlabelled data through contrastive learning and data augmentations. Moreover, [2] propose consistency-diversity augmentations for node and graph classification tasks, and [45] suggest a mixup-based method that considers all available data. Our work differs from the above as follows. First, we utilize an information maximization approach, with a class entropy balancing term in Eq. (9). Second, we propose a total-variation smoothing loss that is adherent to the natural edges of the graph signal, and is shown in Section 5.3 and Table 8 to obtain improved accuracy compared to a Laplacian smoothing regularization as in [51] and a standard total-variation loss that was utilized in GNNs [21, 29, 27, 44]. Third, we propose a cross-validating gradients approach to further improve learning from labelled nodes on top of the standard cross-entropy loss. To the best of our knowledge, the idea of a cross-validation loss through the gradients of the network has not been utilized in GNNs.

2.3 Mutual Information in Neural Networks

The concept of Mutual Information in machine learning tasks was originally used for image and volume alignment by [42]. Recently, it was implemented into CNNs by the seminal Deep InfoMax [20] and into GNNs by [40], where unsupervised learning tasks are considered, by defining some task that is defined by the data (e.g., signal reconstruction) and enforcing information maximization between inputs and their reconstruction or predictions. This concept was found to be useful in a wide array of applications, from image superpixel segmentation [13] to unsupervised graph related tasks [40]. In this paper, we show that this concept significantly improves the overall performance of GNNs when labelled information is partially available, without any architectural changes.

3 Notations and Technical Background

Notations. We now provide the notations that will be used throughout this paper. Let us denote an undirected graph defined by the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of n nodes and $\mathcal{E}$ is a set of m edges. We define the neighbourhood of the i-th node by $\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{E}\}$. Let us denote by $\mathbf{f}^{(l)} \in \mathbb{R}^{n \times c}$ the feature tensor of the nodes $\mathcal{V}$ with c channels at the l-th layer. We denote the adjacency matrix by $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{ij} = 1$ if there exists an edge $(i,j) \in \mathcal{E}$ and 0 otherwise. We also define the diagonal degree matrix $\mathbf{D}$ where $\mathbf{D}_{ii}$ is the degree of the i-th node. We denote the adjacency and degree matrices with added self-loops by $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{D}}$, respectively.

In this paper, we consider the node classification task, where the goal is to assign a class to each node in the graph. We denote the number of classes by k. Since not all nodes are labelled in our considered datasets, we denote the labelled set of vertices by $\mathcal{V}^{lab} \subset \mathcal{V}$, and the one-hot ground-truth labels by $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}^{lab}| \times k}$. To present our approach, which also considers the unlabelled nodes, we denote the node classification prediction tensor by

$$\hat{\mathbf{y}} = \mathrm{SoftMax}(\mathbf{f}^{out}) \in \mathbb{R}^{n \times k}, \tag{1}$$

where $\mathbf{f}^{out}$ is the output last layer in the network.

GNN Backbones. We consider three popular backbones to demonstrate our ENC approach. Namely, we utilize GCN [24], GAT [39], and GCNII [6], as described below.

GCN defines a propagation operator $\tilde{\mathbf{P}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, and its architecture is given by

$$\mathbf{f}^{(l+1)} = \sigma(\tilde{\mathbf{P}}\mathbf{f}^{(l)}\mathbf{W}^{(l)}), \tag{2}$$

where $\mathbf{W}^{(l)}$ is a 1×1 convolution matrix, i.e., a linear layer, and σ is a non-linear activation function.

GAT defines the propagation operator according to the following edge weight:

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(l)^\top}[\tilde{\mathbf{W}}^{(l)}\mathbf{f}_i \,||\, \tilde{\mathbf{W}}^{(l)}\mathbf{f}_j]\right)\right)}{\sum_{p\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(l)^\top}[\tilde{\mathbf{W}}^{(l)}\mathbf{f}_i \,||\, \tilde{\mathbf{W}}^{(l)}\mathbf{f}_p]\right)\right)}, \tag{3}$$

where $\mathbf{a}^{(l)} \in \mathbb{R}^{2c}$ and $\tilde{\mathbf{W}}^{(l)} \in \mathbb{R}^{c\times c}$ are trainable parameters and $||$ denotes channel-wise concatenation.

By gathering $\alpha_{ij}^{(l)}$ for every edge $(i,j) \in \mathcal{E}$ into a propagation matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$, a GAT layer reads:

$$\mathbf{f}^{(l+1)} = \sigma(\mathbf{S}^{(l)}\mathbf{f}^{(l)}\mathbf{W}^{(l)}). \tag{4}$$

GCNII alters the propagation of GCN as follows:

$$\mathbf{S}^{(l)}(\mathbf{f}^{(l)}, \mathbf{f}^{(0)}) = (1-\alpha^{(l)})\tilde{\mathbf{P}}\mathbf{f}^{(l)} + \alpha^{(l)}\mathbf{f}^{(0)}, \tag{5}$$

where $\alpha^{(l)} \in [0,1]$ is hyper-parameter and $\mathbf{f}^{(0)}$ are the features of the opening (embedding) layer. Then, a GCNII layer is given by:

$$\mathbf{f}^{(l+1)} = \sigma(\beta^{(l)}\mathbf{S}^{(l)}(\mathbf{f}^{(l)}, \mathbf{f}^{(0)})\mathbf{W}^{(l)} + (1-\beta^{(l)})\mathbf{S}^{(l)}(\mathbf{f}^{(l)}, \mathbf{f}^{(0)})), \tag{6}$$

where $\beta^{(l)} \in [0,1]$ is a hyper-parameter.

4 Method

The core idea of our method is that unlike the typical training scheme in GNNs, we treat every node in the data, regardless if it is labelled or not. Our experiments in Section 5 show that every node counts to improve performance, and thus we call our method ENC, which is summarized by the (three-term) extended objective function:

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha\mathcal{L}_{MI} + \beta\mathcal{L}_{TV} + \gamma\mathcal{L}_{CVG}, \tag{7}$$

where α , β , γ are non-negative hyper-parameters. Here, $\mathcal{L}_{CE}$ is the standard cross-entropy objective used in node classification tasks:

$$\mathcal{L}_{CE} = -\frac{1}{\mid \mathcal{V}^{lab} \mid}\sum_{i\in\mathcal{V}^{lab}}\sum_{s=1}^{k}\mathbf{y}_{i,s}\log(\hat{\mathbf{y}}_{i,s}), \tag{8}$$

where $\hat{\mathbf{y}}$ is the prediction tensor defined in Eq. (1). By definition, it considers only the labelled nodes $\mathcal{V}^{lab}$.

Overall, our method can potentially be applied to any GNN that is optimized in an end-to-end fashion, as we propose a *objective function* modification, rather than architectural. The overall flow of our method is shown in Fig. 2.

In what follows, we present each of the proposed objectives. Later, in Section 5, we show the empirical contribution of each individual term, as well as the combinations of the terms, in Fig. 3 and Fig. 4. Furthermore, we perform an hyper-parameter study in Appendix A.2. We further note that our method in Eq. (7) does not require any architectural modifications to the baseline GNNs.

4.1 Node-Class Mutual Information

As discussed in Section 2.3, the idea of mutual information maximization was utilized in unsupervised settings to improve results. Here, we will use it to improve the performance of GNNs for semi-supervised tasks. We define the mutual information objective as:

$$\mathcal{L}_{\text{MI}} = \frac{1}{\mid \mathcal{V} \mid}\sum_{i\in\mathcal{V}}\sum_{s=1}^{k} -\hat{\mathbf{y}}_{i,s}\log\hat{\mathbf{y}}_{i,s} + \lambda\sum_{s=1}^{k}\tilde{\mathbf{y}}_s\log\tilde{\mathbf{y}}_s, \tag{9}$$

where $\tilde{\mathbf{y}}_s = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}}\hat{\mathbf{y}}_{i,s}$ measures the mean prediction probability of the s-th class. The first term in Eq. (9) describes the entropy of the class prediction $[\hat{\mathbf{y}}_{i,s}]_{s=1}^{k}$ for each node $i \in \mathcal{V}$ individually, pushing the predicted classification vector to be deterministic. The second term considers the negative entropy of the mean class prediction probability, promoting the class-uniformity of the prediction. Note that $\mathcal{L}_{MI}$ considers all of the nodes $\mathcal{V}$, and in particular, is not dependent on ground-truth labels as the standard cross-entropy loss. The value of λ is a hyper-parameter and is dependent on the dataset, as a higher value promotes the network to equalize the number of predictions per class. In case $\lambda = 1$, Eq. (9) maximizes the mutual information of the node and class features [3]. Unless stated otherwise, in all experiments we set $\lambda = 2$, and in Appendix A.2, we report the obtained accuracy using different values of λ.

4.2 Total-Variation Regularization

The objective $\mathcal{L}_{TV}$ measures the node prediction discrepancy of neighbouring nodes, by utilizing the Total-Variation (TV) [32] anisotropic smoothness prior that promotes correspondence between smooth regions while preserving boundaries in input features of the graph. To this end, we first define the gradient operator of the graph node features $\mathbf{f}$ as:

$$(\nabla_{\mathcal{G}_\mathbf{w}}\mathbf{f})_{ij} = (\mathbf{w}_i\mathbf{f}_i - \mathbf{w}_j\mathbf{f}_j), \tag{10}$$

where nodes i and j are connected via the (i,j)-th edge, and $\mathbf{w} \in \mathbb{R}^{|\mathcal{V}|}$ is a node weight vector, which we discuss further below. Note that $\nabla_{\mathcal{G}_\mathbf{w}}\mathbf{f}$ is a matrix of size $\mid \mathcal{E} \mid \times c$. With this definition, the standard TV regularization term is given by:

$$\frac{1}{\mid \mathcal{E} \mid}\sum_{(i,j)\in\mathcal{E}}\left\|(\nabla_{\mathcal{G}_\mathbf{w}}\hat{\mathbf{y}})_{ij}\right\|_1, \tag{11}$$

where $\hat{\mathbf{y}}$ is the prediction tensor defined in Eq. (1) and $\mathbf{f}^{in} \in \mathbb{R}^{n\times c_{in}}$ are the input features tensor with c_{in} channels. We augment Eq. (11) with an additional term that guides the TV regularization term to adhere to the boundaries in the input node features, as follows:

$$\mathcal{L}_{\text{TV}} = \frac{1}{\mid \mathcal{E} \mid}\sum_{(i,j)\in\mathcal{E}}\left(\left\|(\nabla_{\mathcal{G}_\mathbf{w}}\hat{\mathbf{y}})_{ij}\right\|_1 \exp^{-\|(\nabla_{\mathcal{G}_\mathbf{w}}\mathbf{f}^{in})_{ij}\|_2^2/\sigma}\right), \tag{12}$$

where σ is a scalar, set to 10 in our experiments. The exp term in Eq. (12) was also proposed in the context of unsupervised image semantic segmentation tasks [18].

Total-Variation and the Dirichlet energy. In Eq. (10) we set $\mathbf{w}_i = 1/\sqrt{d_i+1}$, where d_i is the degree of the i-th node, and $\mathbf{f}_i$ and $\mathbf{f}_j$ are the features of the i-th and j-th nodes, respectively. Note that the gradient operator is a mapping from the graph nodes to the edges, i.e., $\nabla_{\mathcal{G}_\mathbf{w}} : \mathcal{V} \longrightarrow \mathcal{E}$, and in particular its ℓ_2 norm coincides with the Dirichlet energy:

$$E(\mathbf{f}) = \sum_{(i,j)\in\mathcal{E}}\frac{1}{2}\left\|\frac{\mathbf{f}_i}{\sqrt{(1+d_i)}} - \frac{\mathbf{f}_j}{\sqrt{(1+d_j)}}\right\|_2^2 = \|\nabla_{\mathcal{G}_\mathbf{w}}\mathbf{f}\|_2^2. \tag{13}$$

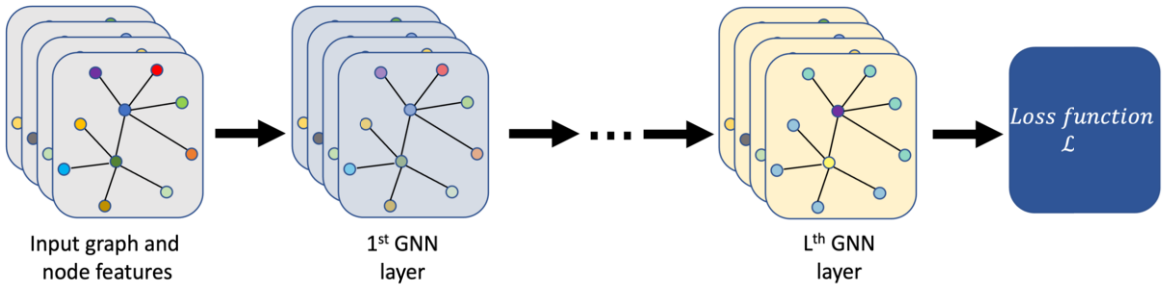


Figure 2: The overall scheme of our method. ENC is a novel loss objective for training GNNs. Given a GNN (e.g., GCN or GAT), we compute our loss defined in Eq. (7), and update the model weights using backpropagation.

This observation uncovers an important nature of the proposed regularization technique — it demands the similarity of the Dirichlet energy between input features and the node classification predictions. Nonetheless, using an ℓ_2 norm to measure similarity is known not to respect the signal boundaries [48], and typically smooth them. We therefore resort to the ℓ_1 norm in Eq. (12), which was shown to be useful for color image processing when boundaries need to be preserved [26]. The concept of preserving the Dirichlet energy is often used to avoid the over-smoothing phenomenon [6, 58] by adding terms to the *network architecture*. Although the focus of this work is to obtain improved training, our experiments in Section 5 show that ENC also somewhat eases over-smoothing, albeit does not prevent it, as the architectures are not changed. Also, we show that compared with other smoothness objective terms like P-reg [51], our Total-Variation loss tends to yield improved results also on heterophilic datasets, as presented and discussed in Section 5.3 and Table 8.

4.3 Cross-Validating Gradients

The components in previous sections consider the inclusion of unlabelled nodes to the optimization objective. In this section we propose a third and final piece of our approach that seeks to improve the learning from the *labelled* nodes. As discussed in Section 1, the typical training of GNNs involves the minimization of the cross-entropy loss of the labelled nodes, as described in Eq. (8). Here we use a mechanism to improve the training by requiring gradient consistency throughout the training process, which we obtain by demanding a cross-validation of the training procedure that we describe now.

Let us denote a random disjoint partition of the labelled nodes indices $p_1,\ p_2 \subset \mathcal{V}^{lab}$ where $p_1 \bigcup p_2 = \mathcal{V}^{lab}$ and $p_1 \bigcap p_2 = \emptyset$, that is uniformly drawn at each iteration during training. Let us consider the partial cross-entropy loss with respect to p_1 and p_2, i.e., $L_{CE}(\mathcal{V}^{lab}_{p_1})$ and $L_{CE}(\mathcal{V}^{lab}_{p_2})$. We wish that a gradient step computed with respect to the nodes p_1 will decrease the objective computed with respect to p_2 and vice versa. If this is not the case then the proposed direction may over-fit to a particular partition of points. This discussion is at the core of the rationale for using Generalized Cross Validation (GCV) [19] designed to prevent over-fitting for the mean square error loss and uses Jacobians with respect to the parameters.

To avoid using Jacobians for the cross-entropy loss, we penalize the gradient steps that point to directions that fit one set of points p_1 and not the other p_2 by measuring their negative cosine similarity:

$$\mathcal{L}_{CVG} = -\frac{g^{p_1}_{\boldsymbol{\theta}} \cdot g^{p_2}_{\boldsymbol{\theta}}}{\|g^{p_1}_{\boldsymbol{\theta}}\| \|g^{p_2}_{\boldsymbol{\theta}}\|}, \tag{14}$$

where $g^{p_1}_{\boldsymbol{\theta}} = \frac{\partial L_{CE}(\mathcal{V}^{lab}_{p_1})}{\partial \boldsymbol{\theta}}$ and $g^{p_2}_{\boldsymbol{\theta}} = \frac{\partial L_{CE}(\mathcal{V}^{lab}_{p_2})}{\partial \boldsymbol{\theta}}$ are the gradients of the partial cross-entropy losses with respect to the weights $\boldsymbol{\theta}$, and $\cdot$ is the dot-product operation. During training, we randomly generate equally-sized $p_1\ , p_2$ using random permutations of the labelled nodes $\mathcal{V}^{lab}$. Each iteration can be thought of as a 2-fold cross-validation and it promotes steps that generalize the fit of the data. We show the positive effect of the stochasticity of $p_1\ , p_2$ in Section 5.3.

4.4 Computational Costs

The losses $\mathcal{L}_{MI}$ and $\mathcal{L}_{TV}$ do not add significant computations, as they only consider the output of the network and perform simple operations. The loss $\mathcal{L}_{CVG}$ requires additional computations as it first computes the gradients $g^{p_1}_{\boldsymbol{\theta}}$ and $g^{p_2}_{\boldsymbol{\theta}}$. We provide run-times and accuracy of the GNN baselines and ENC in Appendix A.3. We can see that our approach offers a significant accuracy improvement over the baselines, at an insignificant additional cost using $\mathcal{L}_{MI}$ and $\mathcal{L}_{TV}$. Using all our losses from Eq. (7) (that is, also including $\mathcal{L}_{CVG}$) requires more computations, yielding further accuracy improvements.

5 Experiments

We demonstrate our ENC on semi- and fully-supervised node classification followed by an ablation study. In all experiments, we consider three popular baselines, namely GCN, GAT, and GCNII, to which we incorporate our ENC. We note that our method is general and can be incorporated into most GNNs, as it requires no architectural changes. A detailed description of the baselines that serve as backbones to our ENC approach is given in Section 3, and the general architecture is provided in Table 1, and further information about the architecture is given in Appendix A.1. We use the Adam optimizer, and perform a grid search to choose the hyper-parameters. The range and selected values are reported in Appendix A.2. Our code is implemented using PyTorch, trained on an Nvidia Titan RTX GPU. The datasets' statistics are provided in Table 2. We show that for all tasks and datasets, ENC offers a consistent improvement over baseline methods. For example, in Table 3, ENC-GCN obtains 85.3% accuracy on Cora vs. 81.2% using the standard GCN, an improvement of 4.1%. Since some baseline methods do not report standard-deviation, we report mean accuracy in the main paper. In Appendix A.4, we provide the mean and standard-deviation accuracy of ENC.

5.1 Semi-Supervised Node Classification

We consider Cora, Citeseer and Pubmed datasets and the standard public training/validation/testing split as in [52], with 20 nodes per class for training. We follow the training and evaluation scheme of [6] and consider various GNN models like GCN, GAT, superGAT

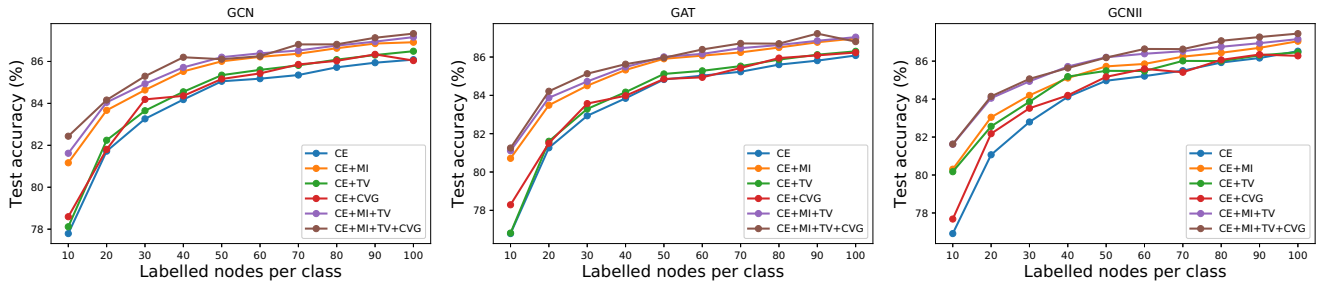


Figure 3: Cora test accuracy (%) with a varying number of labels and 100 random splits per configuration. We compare the influence of the proposed losses. Our ENC approach (all losses together) consistently improves accuracy by an average of 2%.

Table 1: The GNN architecture in our experiments.

Input size	Layer	Output size
$n \times c_{in}$	Dropout(p)	$n \times c_{in}$
$n \times c_{in}$	1×1 Convolution	$n \times c$
$n \times c$	ReLU	$n \times c$
$n \times c$	$L\times$ ENC-GNN layers	$n \times c$
$n \times c$	Dropout(p)	$n \times c$
$n \times c$	1×1 Convolution	$n \times k$

Table 2: Datasets statistics.

Dataset	Classes	Nodes	Edges	Features
Cora	7	2,708	5,429	1,433
Citeseer	6	3,327	4,732	3,703
Pubmed	3	19,717	44,338	500
Chameleon	5	2,277	36,101	2,325
Actor	5	7,600	33,544	932
Squirrel	5	5,201	198,493	2,089
Cornell	5	183	295	1,703
Texas	5	183	309	1,703
Wisconsin	5	251	499	1,703
Ogbn-arxiv	40	169,343	1,166,243	128

[23], APPNP [25], JKNet [49] , GCNII, GRAND [4], PDE-GCN [11] and EGNN[58] and superGAT [23]. Also, we compare our approach with label propagation (LP) and regularization/augmentation based methods. For the former, we consider GCN-LPA [43] and PTA [10]. For the latter, we compare with GAUG [57], DropEdge [31], GraphVAT [14], GRAND [15], P-reg [51], GraphMix [41], NodeAug [47], NASA [2] and local augmentations (LA) by [28]. We summarize the results in Table 3 where we see better or on-par performance with other state-of-the-art methods and a significant increase over the baselines of GCN, GAT, and GCNII. For example, we obtain 83.8% accuracy on Pubmed using our ENC-GCNII compared to 80.3% with GCNII. We also provide a tSNE plot of the node predictions on Cora using GCN and our ENC-GCN in Figure 1. Additionally, we experiment with a varying number of layers, from 2 to 64, and report the results in Table 4. The evaluation of our method with a varying number of layers sheds light on the ability of our method to ease the over-smoothing phenomenon. While our method does not prevent the over-smoothing of the baseline method (as it does not alter the architecture), we can see that compared to the over-smoothing baseline methods GCN and GAT, their counterparts ENC-GCN and ENC-GAT, respectively, show a slower accuracy degradation.

Table 3: Semi-supervised node classification accuracy (%).

Method	Cora	Citeseer	Pubmed
GCN	81.1	70.8	79.0
GAT	83.1	70.8	78.5
GCNII	85.5	73.4	80.3
ChebNet	81.2	69.8	74.4
APPNP	83.3	71.8	80.1
JKNET	81.1	69.8	78.1
GRAND [4]	84.7	73.6	81.0
PDE-GCN	84.3	75.6	80.6
EGNN	85.7	–	80.1
superGAT	84.3	72.6	81.7
GCN-LPA	82.8	72.3	78.6
PTA	83.0	71.6	80.1
GAUG	83.6	73.3	80.2
DropEdge	82.8	72.3	79.6
GraphVAT	82.9	73.8	79.5
GraphMix	84.0	74.7	81.1
P-reg	83.9	74.8	80.1
NodeAug	84.3	74.2	81.5
GRAND [15]	85.4	75.4	82.7
NASA	85.1	75.5	80.2
LA-GCN	84.6	74.7	81.7
LA-GAT	84.7	73.7	81.0
LA-GCNII	85.7	74.1	80.6
LA-GRAND	85.7	75.8	83.4
ENC-GCN (Ours)	85.3	75.5	81.9
ENC-GAT (Ours)	85.2	**75.8**	82.6
ENC-GCNII (Ours)	**86.0**	75.0	**83.8**

5.2 *Fully-Supervised Node Classification*

To further validate the efficacy of our method, we employ fully supervised node classification on 10 datasets. We examine our ENC-GCN, ENC-GAT and ENC-GCNII on Cora, Citeseer, Pubmed, Chameleon, Squirrel, Actor, Cornell, Texas and Wisconsin using the 10 splits from [30] with train/validation/test label split of $48\%, 32\%, 20\%$ respectively, and report their average accuracy. In all experiments, 64 channels are used and a grid search is used to determine the hyper-parameters. To establish a strong baseline, we consider various methods, namely, GCN, GAT, Geom-GCN [30], APPNP, JKNet, WRGAT [38], GCNII, PDE-GCN, DropEdge, H2GCN [59], GGCN [50], MagNet [55], GPRGNN [7], FAGCN [1], GraphCON [33], and GRAFF [17]. For GGCN, we use the results reported in [17], as the splits used in GGCN [50] are different than the splits considered in our experiments. Additionally, we evaluate our ENC using the larger Ogbn-arxiv dataset using the official train/validation/test split in Table 6. To distinguish between homophilic and heterophilic datasets, we report the results of the former in Table 5, and of the latter in

Table 4: Semi-supervised node classification accuracy (%). – indicates not available results.

Dataset	Method	Layers 2	4	8	16	32	64
Cora	GCN	**81.1**	80.4	69.5	64.9	60.3	28.7
	GAT	**83.1**	81.3	74.5	68.2	58.9	34.1
	GCNII	82.2	82.6	84.2	84.6	85.4	**85.5**
	GCN (Drop)	**82.8**	82.0	75.8	75.7	62.5	49.5
	JKNet (Drop)	–	**83.3**	82.6	83.0	82.5	83.2
	GCNII*	80.2	82.3	82.8	83.5	84.9	**85.3**
	PDE-GCN$_D$	82.0	83.6	84.0	84.2	84.3	**84.3**
	EGNN	83.2	–	–	85.4	–	**85.7**
	ENC-GCN	**85.3**	84.4	83.3	80.9	75.4	60.6
	ENC-GAT	**85.2**	84.5	82.5	80.4	75.1	66.3
	ENC-GCNII	84.7	85.0	85.2	85.9	85.9	**86.0**
Citeseer	GCN	**70.8**	67.6	30.2	18.3	25.0	20.0
	GAT	**70.8**	68.6	55.7	31.2	22.0	20.9
	GCNII	68.2	68.8	70.6	72.9	**73.4**	73.4
	GCN (Drop)	**72.3**	70.6	61.4	57.2	41.6	34.4
	JKNet (Drop)	–	72.6	71.8	**72.6**	70.8	72.2
	GCNII*	66.1	66.7	70.6	72.0	**73.2**	73.1
	PDE-GCN$_D$	74.6	75.0	75.2	75.5	**75.6**	75.5
	ENC-GCN	**75.5**	73.9	73.8	73.0	72.1	70.0
	ENC-GAT	**75.8**	72.1	70.1	65.1	52.2	40.1
	ENC-GCNII	72.5	73.4	73.8	74.0	74.6	**75.0**
Pubmed	GCN	**79.0**	76.5	61.2	40.9	22.4	35.3
	GAT	**78.5**	76.4	69.8	50.1	41.0	42.3
	GCNII	78.2	78.8	79.3	**80.2**	79.8	79.7
	GCN (Drop)	**79.6**	79.4	78.1	78.5	77.0	61.5
	JKNet (Drop)	–	78.7	78.7	**79.7**	79.2	78.9
	GCNII*	77.7	78.2	78.8	**80.3**	79.8	80.1
	PDE-GCN$_D$	79.3	**80.6**	80.1	80.4	80.2	80.3
	EGNN	79.2	–	–	80.0	–	**80.1**
	ENC-GCN	**81.9**	81.0	80.1	78.1	77.5	73.3
	ENC-GAT	**82.6**	81.9	81.5	81.6	80.0	80.1
	ENC-GCNII	81.0	81.4	81.8	82.6	83.2	**83.8**

Table 5: Fully-supervised node classification accuracy (%) on *homophilic* datasets.

Method Homophily	Cora 0.81	Cite. 0.80	Pub. 0.74
GCN	85.77	73.68	88.13
GAT	86.37	74.32	87.62
GCNII	88.49	77.08	89.57
Geom-GCN-I	85.19	77.99	90.05
Geom-GCN-P	84.93	75.14	88.09
Geom-GCN-S	85.27	74.71	84.75
APPNP	87.87	76.53	89.40
JKNet (Drop)	87.46	75.96	89.45
WRGAT	88.20	76.81	88.52
GCNII*	88.01	77.13	**90.30**
GGCN	87.95	77.14	89.15
H2GCN	87.87	77.11	89.49
GPRGNN	87.95	77.13	87.54
ENC-GCN (Ours)	89.07	78.15	88.63
ENC-GAT (Ours)	89.05	78.20	88.59
ENC-GCNII (Ours)	**90.01**	**79.34**	89.35

Table 7. We see a significant improvement across all benchmarks and types of datasets compared to the baseline methods of GCN, GAT and GCNII. To measure the homophily score of the different datasets, we follow the definition in [30]. Compared to methods like GraphCON, GGCN, and H2GCN, our method reads better or similar accuracy while offering an the simplicity of keeping the baseline architectures and changing only their objective. For instance, our ENC-GCNII achieves 90.01% accuracy on Cora compared to 87.95% and 87.87% of GGCN and H2GCN. On a heterophilic dataset like Texas, our ENC-GCNII obtains an accuracy of 86.21%, while methods like GraphCON and H2GCN obtain 85.40% and 84.86%.

Table 6: Ogbn-arxiv node classification accuracy (%).

Method	Acc. (%)
GCN	71.74
GAT	71.59
GCNII	72.74
APPNP	71.82
GATv2	71.87
EGNN	72.70
APPNP	72.23
GRAND	72.70
ENC-GCN (Ours)	72.95
ENC-GAT (Ours)	73.38
ENC-GCNII (Ours)	**73.56**

Table 7: Fully-supervised node classification accuracy (%) on *heterophilic* datasets.

Method Homophily	Squirrel 0.22	Actor 0.22	Cham. 0.23	Corn. 0.30	Texas 0.11	Wisc. 0.21
GCN	23.96	26.86	28.18	52.70	52.16	48.92
GAT	30.03	28.45	42.93	54.32	58.38	49.41
GCNII	38.47	32.87	60.61	74.86	69.46	74.12
Geom-GCN-I	38.32	29.09	60.31	56.76	57.58	58.24
Geom-GCN-P	38.14	31.63	60.90	60.81	67.57	64.12
Geom-GCN-S	36.24	30.30	59.96	55.68	59.73	56.67
JKNet (Drop)	35.93	29.54	62.08	61.08	57.30	50.59
PairNorm	50.44	27.40	62.74	58.92	60.27	48.43
GCNII*	39.92	33.61	62.48	76.49	77.84	81.57
GRAND	40.05	35.62	54.67	82.16	75.68	79.41
WRGAT	48.85	36.53	65.24	81.62	83.62	86.98
MagNet	–	–	–	84.30	83.30	85.70
GGCN	55.17	**37.81**	71.14	85.68	84.86	86.86
H2GCN	36.48	35.70	60.11	82.70	84.86	87.65
GPRGNN	31.61	34.63	46.58	80.27	78.38	82.94
FAGCN	42.59	34.87	55.22	79.19	82.43	82.94
GraphCON-GCN	–	–	–	84.30	85.40	87.80
GraphCON-GAT	–	–	–	83.20	82.20	85.70
GRAFF	**59.01**	37.11	**71.38**	84.05	**88.38**	**88.83**
ENC-GCN (Ours)	51.81	31.89	58.72	73.14	61.08	59.80
ENC-GAT (Ours)	46.77	32.71	60.41	76.95	69.72	64.31
ENC-GCNII (Ours)	54.20	34.82	66.32	**88.38**	86.21	87.84

5.3 Ablation Study

Influence of the objectives As our ENC objective is comprised of several objectives, it is important to delve into their contribution, individually and jointly, under different settings. We again use the GCN, GAT and GCNII architectures, and the Cora dataset. For a comprehensive study, we vary the number of labelled nodes per class from 10 to 100, with intervals of 10, and report the obtained test accuracy. To ensure statistically meaningful results and following observations from [35] regarding the evaluation of GNNs, for each experiment we report the average accuracy of 100 random splits with the respective number of labelled nodes. We present the results in Figure 3, where we can see that all of our objectives positively contribute to the obtained accuracy compared with the baseline case of using cross-entropy loss only. That is, we see that our ENC approach presented in Eq. (7) obtains the best results across all considered set-

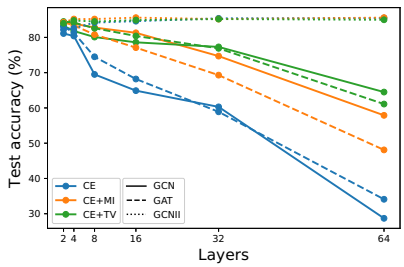


Figure 4: MI vs. TV loss on Cora using GCN, GAT, GCNII with a variable number of layers. TV loss obtains the best results with respect to the network's depth.

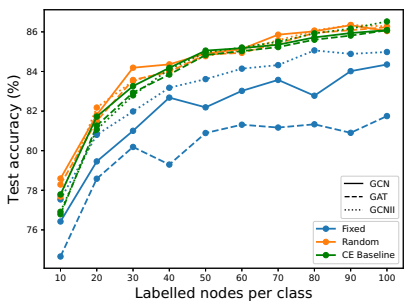


Figure 5: The effect of randomly sampling p_1 and p_2 demonstrated on the Cora dataset using GCN, GAT and GCNII. Compared with CE as baseline, CVG is most helpful in low-labelled data regime.

tings. Also, while ENC does not prevent over-smoothing, it is observed from Table 4 that it can somewhat ease over-smoothing, while not changing the architecture, but only the loss function. To further investigate where this property stems from, we examined the performance on the public split of Cora with each of the proposed losses. We found that the $\mathcal{L}_{MI}$ and $\mathcal{L}_{TV}$ improve the performance of deep networks based on the GCN and GAT baselines, which are known to be over-smoothing [5, 56]. Specifically, we find that the TV loss yields the best results as an individual loss with respect to the depth of the networks. We found that using the CVG loss does not achieve a similar effect. For a complete comparison, we also report the results with GCNII as a baseline. The results are reported in Figure 4.

Fixed vs. Random nodes partition We study the effect of randomly partitioning the set of labelled nodes $\mathcal{V}^{lab}$ to p_1 , p_2 at every iteration compared to fixing p_1 p_2 throughout the training stage on the Cora dataset, using GCN, GAT and GCNII. In the case of the latter, we report the average accuracy obtained by 10 random initializations of p_1 and p_2 to ensure the significance of the results. We present the results in Figure 5. We can immediately see a large performance gap between the two choices, leading us to employ the random sampling of the node partition at every iteration in our experiments.

TV vs. P-reg terms The P-reg [51] regularization leverages on the known Laplacian regularization [36], to improve the training of GNNs by promoting smooth node predictions. While it was demonstrated to be effective on homophilic datasets like Cora, Citeseer and Pubmed, it is interesting to find whether such a strategy can be beneficial in heterophilic dataets like Cornell, Texas and Wisconsin. Intuitively, one may expect such a method to perform worse, as it demands the similarity of predicted labels and their smoothing by the normalized adjacency matrix, as follows:

$$\mathcal{L}_{P-reg} = \|\hat{\mathbf{y}} - \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\hat{\mathbf{y}}\|_2^2. \tag{15}$$

In contrast, our TV regularization in Eq. (12) suggests to promote learning predictions that are both smooth and adherent to the natural boundaries of the input features. We therefore expect that for heterophilic datasets, our method will perform better. To this end we experiment with GCN joint with each of the regularizations, one at a time, and report the test accuracy obtained on Cora, Citeseer and Pubmed, Cornell, Texas and Wisconsin using the splits from [30] in Table 8. We find that for homophilic datasets, both methods improve the baseline GCN. However, for heterophilic datasets, P-reg regularization can harm the accuracy, further highlighting the contribution of TV as a smoothing but also boundary-preserving regularizer.

Table 8: TV vs. P-reg regularization applied to GCN. Metric is accuracy (%). Hom. denotes Homophily. Our TV regularization (Eq. (12)) improves accuracy on homophilic and heterophilic datasets.

Dataset	Hom.	GCN	GCN+ Eq. (11)	GCN+ P-reg.	GCN+ Eq. (12) (Ours)
Cora	0.81	85.77	87.47	87.42	**88.17**
Citeseer	0.80	73.68	77.02	76.96	**77.10**
Pubmed	0.74	88.13	87.14	**88.34**	**88.34**
Cornell	0.30	52.70	63.22	61.25	**66.71**
Texas	0.11	52.16	56.80	50.21	**58.16**
Wisconsin	0.21	48.92	57.83	47.88	**58.39**

6 Conclusion

In this paper, we propose an orthogonal path to the recent advances in GNNs. While most methods focus on improving GNN architectures and relying on the standard cross-entropy loss, we show that by including our ENC objectives in the optimization process leads to major improvements of baseline methods like GCN, GAT and GCNII. Our method often achieves better or similar results to other state-of-the-art methods that are more complex. We motivate our objectives by adapting knowledge and concepts from fields like Computer Vision, Image Processing and Optimization methods that are often found in CNNs but not in GNNs, and validate their efficacy in our extensive set of experiments. Our method is general, and we deem that it will also be beneficial for training future GNN architectures.

References

[1] D. Bo, X. Wang, C. Shi, and H. Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, 2021.

[2] D. Bo, B. Hu, X. Wang, Z. Zhang, C. Shi, and J. Zhou. Regularizing graph neural networks via consistency-diversity graph augmentations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3913–3921, 2022.

[3] J. Bridle, A. Heading, and D. MacKay. Unsupervised classifiers, mutual information and'phantom targets. *Advances in neural information processing systems*, 4, 1991.

[4] B. P. Chamberlain, J. Rowbottom, M. Gorinova, S. Webb, E. Rossi, and M. M. Bronstein. Grand: Graph neural diffusion. *arXiv preprint arXiv:2106.10934*, 2021.

[5] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:3438–3445, 04 2020. doi: 10.1609/aaai.v34i04.5747.

[6] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*.

[7] E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[8] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[9] T. H. Do, D. M. Nguyen, G. Bekoulis, A. Munteanu, and N. Deligiannis. Graph convolutional neural networks with node transition probability-based message passing and dropnode regularization. *Expert Systems with Applications*, 174:114711, 2021.

[10] H. Dong, J. Chen, F. Feng, X. He, S. Bi, Z. Ding, and P. Cui. On the equivalence of decoupled graph convolution network and label propagation. In *Proceedings of the Web Conference 2021*, 2021.

[11] M. Eliasof, E. Haber, and E. Treister. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] M. Eliasof, E. Haber, and E. Treister. Every node counts: Improving the training of graph neural networks on node classification. *arXiv preprint arXiv:2211.16631*, 2022.

[13] M. Eliasof, N. B. Zikri, and E. Treister. Rethinking unsupervised neural superpixel segmentation. *arXiv preprint arXiv:2206.10213*, 2022.

[14] F. Feng, X. He, J. Tang, and T.-S. Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2493–2504, 2019.

[15] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33, 2020.

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[17] F. D. Giovanni, J. Rowbottom, B. P. Chamberlain, T. Markovich, and M. M. Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[18] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[19] G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2): 215–223, 1979.

[20] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[21] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[22] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[23] D. Kim and A. Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2020.

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

[25] J. Klicpera, A. Bojchevski, and S. Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019.

[26] D. Kong, R. Fujimaki, J. Liu, F. Nie, and C. Ding. Exclusive feature learning on arbitrary structures via $\ell_{1,2}$ -norm. *Advances in neural information processing systems*, 27, 2014.

[27] P. Liao, H. Zhao, K. Xu, T. Jaakkola, G. J. Gordon, S. Jegelka, and R. Salakhutdinov. Information obfuscation of graph neural networks. In *International Conference on Machine Learning*. PMLR, 2021.

[28] S. Liu, R. Ying, H. Dong, L. Li, T. Xu, Y. Rong, P. Zhao, J. Huang, and D. Wu. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pages 14054–14072. PMLR, 2022.

[29] X. Liu, W. Jin, Y. Ma, Y. Li, H. Liu, Y. Wang, M. Yan, and J. Tang. Elastic graph neural networks. In *International Conference on Machine Learning*, pages 6837–6849. PMLR, 2021.

[30] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[31] Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

[32] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 1992.

[33] T. K. Rusch, B. Chamberlain, J. Rowbottom, S. Mishra, and M. Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022.

[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[35] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

[36] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003.

[37] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

[38] S. Suresh, V. Budde, J. Neville, P. Li, and J. Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[40] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

[41] V. Verma, M. Qu, K. Kawaguchi, A. Lamb, Y. Bengio, J. Kannala, and J. Tang. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10024–10032, 2021.

[42] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International journal of computer vision*, 24(2), 1997.

[43] H. Wang and J. Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.

[44] X. Wang, H. Liu, C. Shi, and C. Yang. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, 34:23768–23779, 2021.

[45] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*.

[46] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[47] Y. Wang, W. Wang, Y. Liang, Y. Cai, J. Liu, and B. Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining (KDD)*, 2020.

[48] J. Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[49] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 2018.

[50] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, and D. Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*, 2021.

[51] H. Yang, K. Ma, and J. Cheng. Rethinking graph regularization for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4573–4581, 2021.

[52] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[53] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[54] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[55] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. Hirn. Magnet: A neural network for directed graphs. *Advances in Neural Information Processing Systems*, 34:27003–27015, 2021.

[56] L. Zhao and L. Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

[57] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah. Data augmentation for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

[58] K. Zhou, X. Huang, D. Zha, R. Chen, L. Li, S.-H. Choi, and X. Hu. Dirichlet energy constrained learning for deep graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[59] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*.