GraphMix: Improved Training of Graph Neural Networks for Semi-Supervised Learning

Vikas Verma [1 2] Meng Qu [2 3] Alex Lamb [2 3] Yoshua Bengio [2 3] Juho Kannala [1] Jian Tang [2 4]

Abstract

We present *GraphMix*, a regularized training scheme for Graph Neural Network based semi-supervised object classification, leveraging the recent advances in the regularization of classical deep neural networks. Specifically, we propose a unified approach in which we train a fully-connected network jointly with the graph neural network via parameter sharing, interpolation-based regularization and self-predicted-targets. Our proposed method is architecture agnostic in the sense that it can be applied to any variant of graph neural networks which applies a parametric transformation to the features of the graph nodes. Despite its simplicity, with GraphMix we can consistently improve results and achieve or closely match state-of-the-art performance using even simpler architectures such as Graph Convolutional Networks, across three established graph benchmarks: Cora, Citeseer and Pubmed citation network datasets, as well as three newly proposed datasets :Cora-Full, Co-author-CS and Co-author-Physics.

1. Introduction

Due to the presence of graph-structured data across a wide variety of domains, such as biological networks, citation networks and social networks, there have been several attempts to design neural networks, called graph neural networks (GNN), that can process arbitrarily structured graphs. Early work includes (Gori et al., 2005; Scarselli et al., 2009) which propose a neural network that can directly process most types of graphs e.g., acyclic, cyclic, directed, and undirected graphs. More recent approaches include (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Kipf & Welling, 2016; Gilmer et al., 2017; Hamilton et al., 2017; Veličković et al., 2018; 2019; Qu et al., 2019; Gao & Ji, 2019; Ma et al., 2019), among others. Many of these approaches are designed for addressing the problem of semi-supervised learning over graph-structured data (Zhou et al., 2018). Much of these research efforts have been dedicated to developing novel architectures.

Here we instead re-synthesize ideas from recent advances in regularizing classical neural network, and propose an architecture-agnostic framework for regularized training of GNN-based semi-supervised node and edge classification. Recently, regularization based on data-augmentation has been shown to be very effective in other types of neural networks but how to apply these techniques in GNNs is still under-explored. Our proposed method GraphMix [1] is a unified framework that utilizes interpolation based data augmentation (Zhang et al., 2018; Verma et al., 2019a) and self-training based data-augmentation (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Verma et al., 2019b; Berthelot et al., 2019). Our method offers the following advantages:

- With this method we can achieve state-of-the-art performance even when using simpler GNN architectures such as Graph Convolutional Networks (Kipf & Welling, 2017) (Section 4.2).
- Through thorough experiments we show that the method improves test accuracy over a number of underlying GNN architectures and datasets (Section 4.2, 4.3 and 4.4).
- The method has no additional memory requirement and negligible additional computational cost. It can be implemented on top of an existing GNN using only a few lines of code (Section 3.2.3).
- It improves the test accuracy over the underlying GNNs with minimal hyperparameter search, thus not being very sensitive to the hyperparameters (Appendix A.5).

[1]Aalto University, Finland [2]Mila - Québec Artificial Intelligence Institute, Montréal, Canada [3]Universite de Montreal, Canada [4]HEC, Montréal, Canada. Correspondence to: Vikas Verma <vikasverma.iitm@gmail.com>.

[1]code available at https://github.com/vikasverma1077/GraphMix

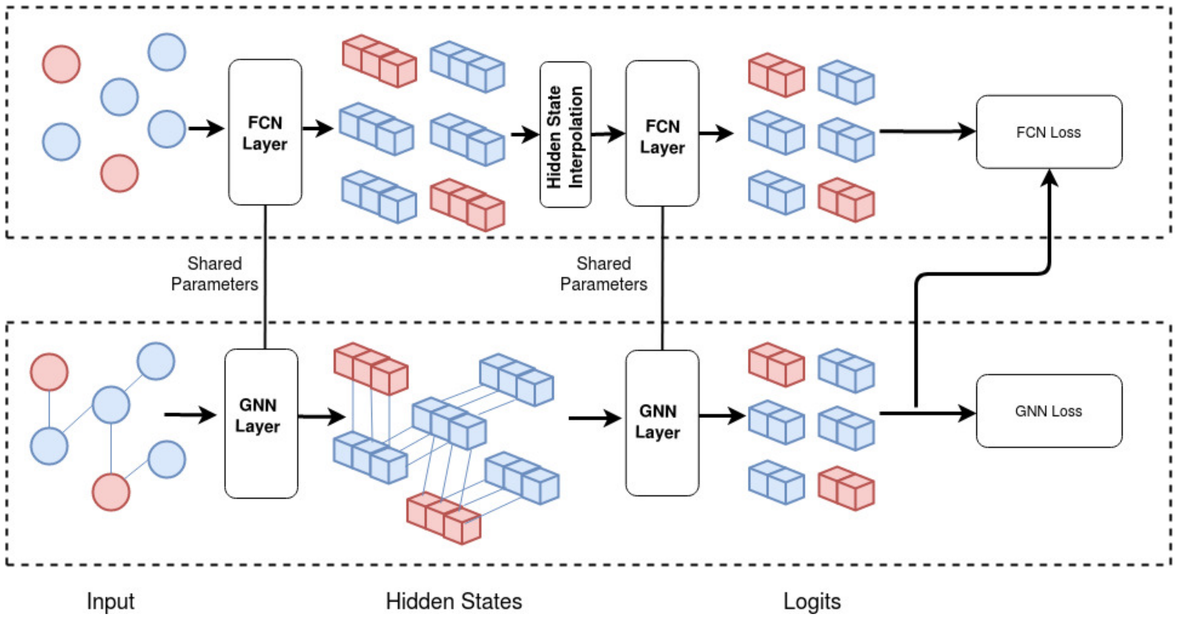


Figure 1. The procedure for training with GraphMix . The labeled and unlabeled nodes are shown with different colors in the graph. GraphMix augments the training of a baseline Graph Neural Network (GNN) with a Fully-Connected Network (FCN). The FCN is trained by interpolating the hidden states and the corresponding labels. This leads to better features which are transferred to the GNN via sharing the linear transformation parameters W (in Equation1) of the GNN and FCN layers. Furthermore, the predictions made by the GNN for unlabeled data are used to augment the input data for the FCN. The FCN and the GNN losses are minimized jointly by alternate minimization.

2. Problem Definition and Preliminaries

2.1. Problem Setup

We are interested in the problem of semi-supervised node and edge classification using graph-structured data. We can formally define such graph-structured data as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{X})$, where $\mathcal{V} = \mathcal{V}_l \cup \mathcal{V}_u$ is the union of labeled ($\mathcal{V}_l$) and unlabeled ($\mathcal{V}_u$) nodes in the graph with cardinalities n_l and n_u, and $\mathcal{A}$ is the adjacency matrix representing the edges between the nodes of $\mathcal{V}$, $\mathcal{X} \in \mathbb{R}^{(n_l+n_u)\times d}$ is the input node features. Each node v belongs to one out of C classes and can be labeled with a C-dimensional one-hot vector $y_v \in \mathbb{R}^C$. Given the labels $Y_l \in \mathbb{R}^{n_l \times C}$ of the labeled nodes $\mathcal{V}_l$, the task is to predict the labels $Y_u \in \mathbb{R}^{n_u \times C}$ of the unlabeled nodes $\mathcal{V}_u$.

2.2. Graph Neural Networks

Graph Neural Networks (GNN) learn the l_{th} layer representations of a sample i by leveraging the representations of the samples $NB(i)$ in the neighbourhood of i. This is done by using an aggregation function that takes as an input the representations of all the samples along with the graph structure and outputs the aggregated representation. The aggregation function can be defined using the Graph Convolution layer (Kipf & Welling, 2017), Graph Attention Layer (Veličković et al., 2018), or any general message passing layer (Gilmer et al., 2017). Formally, let $\mathbf{h}^{(l)} \in \mathbb{R}^{n\times k}$ be a matrix containing the k-dimensional representation of n nodes in the l_{th} layer, then:

$$\mathbf{h}^{(l+1)} = \sigma(AGGREGATE(\mathbf{h}^{(l)}\mathbf{W}, \mathcal{A})) \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{k\times k'}$ is a linear transformation matrix, k' is the dimension of $(l+1)_{th}$ layer, $AGGREGATE$ is the aggregation function that utilizes the graph adjacency matrix $\mathcal{A}$ to aggregate the hidden representations of neighbouring nodes and σ is a non-linear activation function, e.g. ReLU.

2.3. Interpolation Based Regularization Techniques

Recently, interpolation-based techniques have been proposed for regularizing neural networks. We briefly describe some of these techniques here. Mixup (Zhang et al., 2018) trains a neural network on the convex combination of input and targets, whereas Manifold Mixup (Verma et al., 2019a) trains a neural network on the convex combination of the hidden states of a randomly chosen hidden layer and the targets. While Mixup regularizes a neural network by enforcing that the model output should change linearly in between the examples in the input space, Manifold Mixup regularizes the neural network by learning better (more discriminative) hidden states.

Formally, suppose $T_\theta(\mathbf{x}) = (f \circ g)_\theta(\mathbf{x})$ is a neural network parametrized with θ such that $g : \mathbf{x} \to \mathbf{h}$ is a function that maps input sample to hidden states, $f : \mathbf{h} \to \hat{\mathbf{y}}$ is a function that maps hidden states to predicted output, λ is a random variable drawn from $\mathrm{Beta}(\alpha, \alpha)$ distribution, $\mathrm{Mix}_\lambda(\mathbf{a}, \mathbf{b}) = \lambda * \mathbf{a} + (1-\lambda) * \mathbf{b}$ is an interpolation function, $\mathcal{D}$ is the data distribution, $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ is a pair of labeled examples sampled from distribution $\mathcal{D}$ and ℓ be a loss function such as cross-entropy loss, then the Manifold Mixup Loss is defined as:

$$\mathcal{L}_{\mathrm{MM}}(\mathcal{D}, T_\theta, \alpha) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \; \mathop{\mathbb{E}}_{(\mathbf{x}',\mathbf{y}')\sim\mathcal{D}} \; \mathop{\mathbb{E}}_{\lambda\sim\mathrm{Beta}(\alpha,\alpha)} \ell(f(\mathrm{Mix}_\lambda(g(\mathbf{x}), g(\mathbf{x}'))), \mathrm{Mix}_\lambda(\mathbf{y}, \mathbf{y}')). \quad (2)$$

We use above Manifold Mixup loss for training an auxiliary Fully-connected-network as described in Section 3.

3. GraphMix

3.1. Motivation

Data Augmentation is one of the simplest and most efficient technique for regularizing a neural network. In the domains of computer vision, speech and natural language, there exist efficient data augmentation techniques, for example, random cropping, translation or Cutout (Devries & Taylor, 2017) for computer vision, (Ko et al., 2015) and (Park et al., 2019) for speech and (Xie et al., 2017) for natural language. However, data augmentation for the graph-structured data remains under-explored. There exists some recent work along these lines but the prohibitive computation cost (see Section 5.3) introduced by these methods make them impractical for real-world large graph datasets. Based on these limitations, our main objective is to propose an efficient data augmentation technique for graph datasets.

Recent work based on interpolation-based data augmentation (Zhang et al., 2018; Verma et al., 2019a) has seen sizable improvements in regularization performance across a number of tasks. However, these techniques are not directly applicable to graphs for an important reason: *Although we can create additional nodes by interpolating the features and corresponding labels, it remains unclear how these new nodes must be connected to the original nodes via synthetic edges such that the structure of the whole graph is preserved.* To alleviate this issue, we propose to train an auxiliary Fully-Connected Network (FCN) using Manifold Mixup as discussed in Section 3.2. Note that the FCN only uses the node features (not the graph structure), thus the Manifold mixup loss in Eq. 2 can be directly used for training the FCN.

Interpolation based data-augmentation techniques have an added advantage for training GNNs. A vanilla GNN learns the representation of each node by iteratively aggregating information from the neighbors of that node (Equation 1). However, this induces the problem of *oversmoothing* (Li et al., 2018; Xu et al., 2018) while training GNNs with many layers. Due to this limitation, GNNs are trained only with a few layers, and thus they can only leverage the local neighbourhood of each node for learning its representations, without leveraging the representations of the nodes which are multiple hops away in the graph. This limitation can be addressed using the interpolation-based method such as Manifold Mixup: in Manifold Mixup, the representations of a randomly chosen pair of nodes is used to facilitate better representation learning; it is possible that the randomly chosen pair of nodes will not be in the local neighbourhood of each other.

Furthermore, drawing inspiration from the success of self-training semi-supervised learning algorithms (which can be interpreted as a form of data-augmentation) (Verma et al., 2019b; Berthelot et al., 2019), we explore self-training and Manifold Mixup based node augmentation in the context of GNNs. Based on these challenges and motivations we present our proposed approach GraphMix for training GNNs in the following Section.

3.2. Method

We first describe GraphMix at a high-level and then give a more formal description. GraphMix augments the vanilla GNN with a Fully-Connected Network (FCN). The FCN loss is computed using *Manifold Mixup* as discussed below and the GNN loss is computed normally. *Manifold Mixup* training of FCN facilitates learning more discriminative node representations (Verma et al., 2019a). An important question is how these more discriminative node representations can be transferred to the GNN? One potential approach could involve maximizing the mutual information between the hidden states of the FCN and the GNN using formulations similar to those proposed by (Hjelm et al., 2019; Sun et al., 2020). However, this requires optimizing additional network parameters. Instead, we propose parameter sharing between FCN and GNN to facilitate the transfer of discriminative node representations from the FCN to the GNN. It is a viable option because as mentioned in Eq 1, a GNN layer typically performs an additional operation ($AGGREGATE$) on the linear transformations of node representations (which are essentially pre-activation representations of the FCN layer). Using the more discriminative representations of the nodes from FCN, as well as the graph structure, the GNN loss is computed in the usual way to further refine the node representations. In this way we can exploit the improved representations from *Manifold Mixup* for training GNNs.

Additionally, we propose to use the predicted targets from the GNN to augment the training set of the FCN. In this way, both the FCN and the GNN facilitate each other's learning process. Both the FCN loss and the GNN loss are optimized in an alternating fashion during training. At inference time, predictions are made using only the GNN.

The GraphMix procedure is highly flexible: it can be applied to any underlying GNN as long as the underlying GNN applies parametric transformations to the node features. In our experiments, we show the improvements over GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2018) and Graph U-Nets (Gao & Ji, 2019) using GraphMix . A diagram illustrating GraphMix is presented in Figure 1 and the full algorithm is presented in Appendix A.1. Further, we draw similarities and difference of GraphMix w.r.t. Co-training framework in the Section 3.2.2 and discuss the memory and computation requirements in Section 3.2.3.

So far we have presented the general design of GraphMix , now we present the GraphMix procedure more formally. Given a graph $\mathcal{G}$, let $(\mathbf{X}_l, \mathbf{Y}_l)$ be the input features and the labels of the labeled nodes $\mathcal{V}_l$ and let $(\mathbf{X}_u)$ be the input features of the unlabeled nodes $\mathcal{V}_u$. Let F_θ and G_θ be a FCN and a GNN respectively, which share the parameters θ. The FCN loss from the labeled data is computed using Eq. 2 as follows:

$$\mathcal{L}_{\text{supervised}} = \mathcal{L}_{\text{MM}}((\mathbf{X}_l, \mathbf{Y}_l), F_\theta, \alpha) \tag{3}$$

For unlabeled nodes $\mathcal{V}_u$, we compute the prediction $\hat{\mathbf{Y}}_{\mathbf{u}}$ using the GNN:

$$\hat{\mathbf{Y}}_{\mathbf{u}} = G_\theta(\mathbf{X}_u) \tag{4}$$

We note that recent state-of-the-art semi-supervised learning methods use a *teacher* model to accurately predict targets for the unlabeled data. The teacher model can be realized as a temporal ensemble of the *student* model (the model being trained) (Laine & Aila, 2016) or by using an Exponential Moving Average (EMA) of the parameters of the student model (Tarvainen & Valpola, 2017; Verma et al., 2019b). Different from these approaches, we use the GNN as a teacher model for predicting the targets for the FCN. This is due to the fact that GNNs leverage graph structure, which in practice, allows them to make more accurate predictions than the temporal ensemble or EMA ensemble of FCN. (although there is no theoretical guarantee for this).

Moreover, to improve the accuracy of the predicted targets in Eq 4, we applied the average of the model prediction on K random perturbations of an input sample along with sharpening as discussed in Section 3.2.1.

Using the predicted targets for unlabeled nodes, we create a new training set $(\mathbf{X}_u, \hat{\mathbf{Y}}_{\mathbf{u}})$. The loss from the unlabeled data for the FCN is computed as:

$$\mathcal{L}_{\text{unsupervised}} = \mathcal{L}_{\text{MM}}((\mathbf{X}_u, \hat{\mathbf{Y}}_u), F_\theta, \alpha) \tag{5}$$

Total loss for training the FCN is given as the weighted sum of above two loss terms.

$$\mathcal{L}_{\text{FCN}} = \mathcal{L}_{\text{supervised}} + w(t) * \mathcal{L}_{\text{unsupervised}} \tag{6}$$

where $w(t)$ is a sigmoid ramp-up function (Tarvainen & Valpola, 2017) which increases from zero to its max value γ during the course of training.

Now let us assume that the loss for an underlying GNN is $\mathcal{L}_{\text{GNN}} = \ell(G_\theta(\mathbf{X}_l), \mathbf{Y}_l)$; the overall GraphMix loss for the joint training of the FCN and GNN can be defined as the weighted sum of the GNN loss and the FCN loss:

$$\mathcal{L}_{\text{GraphMix}} = \mathcal{L}_{\text{GNN}} + \lambda * \mathcal{L}_{\text{FCN}} \tag{7}$$

However, throughout our experiments, optimizing FCN loss and GNN loss alternatively at each training epoch achieved better test accuracy (discussed in Appendix A.9). This has an added benefit that it removes the need to tune weighing hyper-parameter λ.

For *Manifold Mixup* training of FCN, we apply *mixup* only in the hidden layer. Note that in (Verma et al., 2019a), the authors recommended applying mixing in a randomly chosen layer (which also includes the input layer) at each training update. However, we observed under-fitting when applying *mixup* randomly at the input layer or the hidden layer. Applying *mixup* only in the input layer also resulted in underfitting and did not improve test accuracy.

3.2.1. Accurate Target Prediction for Unlabeled Data

A recently proposed method for accurate target predictions for unlabeled data uses the average of predicted targets across K random augmentations of the input sample (Berthelot et al., 2019). In GraphMix we follow this approach: computing the predicted-targets as the average of predictions made by GNN on K drop-out versions of the input sample.

Many recent semi-supervised learning algorithms (Laine & Aila, 2016; Miyato et al., 2018; Tarvainen & Valpola, 2017; Verma et al., 2019b) are based on the cluster assumption (Chapelle et al., 2010), which posits that the class boundary should pass through the low-density regions of the marginal data distribution. One way to enforce this assumption is to explicitly minimize the entropy of the model's predictions $p(y|\mathbf{x}, \theta)$ on unlabeled data by adding an extra loss term to the original loss term (Grandvalet & Bengio, 2005). The entropy minimization can be also achieved implicitly by modifying the model's prediction on the unlabeled data such that the prediction has low entropy and using these low-entropy predictions as targets for the further training of the model. Examples include "Pseudolabels" (Lee, 2013) and "Sharpening" (Berthelot et al., 2019). Pseudolabeling constructs hard (one-hot) labels for the unlabeled samples which have "high-confidence predictions". Since many of the unlabeled samples may have "low-confidence predictions", they can not be used in the Pseudolabeling technique. On the other hand, Sharpening does not require "high-confidence predictions" , and thus it can be used for all the unlabelled samples. Hence in this work, we use Sharpening for entropy minimization. The Sharpening function over the model prediction $p(y|\mathbf{x}, \theta)$ can be formally defined as follows (Berthelot et al., 2019), where T is the temperature hyperparameter and C is the number of classes:

$$\text{Sharpen}(p_i, T) := p_i^{\frac{1}{T}} \bigg/ \sum_{j=1}^{C} p_j^{\frac{1}{T}} \tag{8}$$

3.2.2. Connection to Co-training

The GraphMix approach can be seen as a special instance of the Co-training framework (Blum & Mitchell, 1998). Co-training assumes that the description of an example can be partitioned into two *distinct* views and either of these views would be sufficient for learning given sufficient labeled data. In this framework, two learning algorithms are trained separately on each view and then the prediction of each learning algorithm on the unlabeled data is used to enlarge the training set of the other. Our method has some important differences and similarities to the Co-training framework. Similar to Co-training, we train two neural networks and the predictions from the GNN are used to enlarge the training set of the FCN. An important difference is that instead of using the predictions from the FCN to enlarge the training set for the GNN, we employ parameter sharing for passing the learned information from the FCN to the GNN. In our experiments, directly using the predictions of the FCN for GNN training resulted in reduced accuracy. This is due to the fact that the number of labeled samples for training the FCN is sufficiently low and hence the FCN does not make accurate enough predictions. Another important difference is that unlike the co-training framework, the FCN and GNN do not use completely distinct views of the data: the FCN uses feature vectors $\mathcal{X}$ and the GNN uses the feature vector and adjacency matrix $(\mathcal{X}, \mathcal{A})$.

3.2.3. Memory and Computational Requirements

One of the major limitations of current GNNs, which prohibits their application to real-world large datasets, is their memory complexity. For example, the fastest implementation of GCN, which stores the entire adjacency matrix $\mathcal{A}$ in the memory, has the memory complexity $O(|\mathcal{V}|^2)$. Implementations with lower memory requirements are possible but they incur higher latency cost due to repeatedly loading parts of adjacency matrix in the memory. Due to these reasons, methods which have additional memory requirement in comparison to the baseline GNNs, are less appealing in practice. In GraphMix , since the parameters of the FCN and GNN are shared, there is no additional memory requirement in comparison to the baseline GNNs. Furthermore, GraphMix does not add any *significant computation cost* over the underlying GNN, because the underlying GNN is trained in the standard way and the FCN training requires trivial additional computation cost for computing the predicted-targets (Section 3.2.1) and the interpolation function ($\text{Mix}_\lambda(\mathbf{a}, \mathbf{b})$ in Section 2.3).

4. Experiments

We present results for the GraphMix algorithm using standard benchmark datasets and the standard architecture in Section 4.2 and 4.4. We also conduct an ablation study on GraphMix in Section 4.5 to understand the contribution of various components to its performance. Refer to Appendix A.5 for implementation and hyperparameter details.

4.1. Datasets

We use three standard benchmark citation network datasets for semi-supervised node classification, namely Cora, Citeseer and Pubmed. In all these datasets, nodes correspond to documents and edges correspond to citations. Node features correspond to the bag-of-words representation of the document. Each node belongs to one of C classes. During training, the algorithm has access to the feature vectors and edge connectivity of all the nodes but has access to the class labels of only a few of the nodes.

For semi-supervised link classification, we use two datasets Bitcoin Alpha and Bitcoin OTC from (Kumar et al., 2016; 2018). The nodes in these datasets correspond to the bitcoin users and the edge weights between them correspond to the degree of trust between the users. Following (Qu et al., 2019), we treat edges with weights greater than 3 as positive instances, and edges with weights less than -3 are treated as negative ones. Given a few labeled edges, the task is to predict the labels of the remaining edges. The statistics of these datasets as well as the number of training/validation/test nodes is presented in Appendix A.2.

4.2. Semi-supervised Node Classification

For baselines, we choose GCN (Kipf & Welling, 2017), and the recent state-of-the-art methods GAT (Veličković et al., 2018), GMNN (Qu et al., 2019) and Graph U-Net (Gao & Ji, 2019). We additionally use two self-training based baselines: in the first one, we trained a GCN with self-generated predicted targets, and in the second one, we trained a FCN with self-generated predicted targets, named "GCN (self-training)" and "FCN (self-training)" respectively in Table 1. For generating the predicted-targets in above two baselines, we followed the procedure of Section 3.2.1.

GraphMix(GCN) , GraphMix(GAT) and GraphMix(Graph U-Net) refer to the methods where underlying GNNs are GCN, GAT and Graph U-Net respectively. Refer to Appendix Section A.5 for implementation and hyperparameter details.

(Shchur et al., 2018) demonstrated that the performance of the current state-of-the-art Graph Neural Networks on the standard train/validation/test split of the popular benchmark datasets (such as Cora, Citeseer, Pubmed, etc) is significantly different from their performance on the random splits. For fair evaluation, they recommend using multiple random partitions of the datasets. Along these lines, we created 10 random splits of the Cora, Citeseer and Pubmed with the same train/ validation/test number of samples as in the standard split. We also provide the results for the standard train/validation/test split.

The results are presented in Table 1. We observe that GraphMix always improves the accuracy of the underlying GNNs such as GCN, GAT and Graph U-Net across all the dataset, with GraphMix(GCN) achieving the best results.

We further present results with fewer labeled samples in Appendix Section A.4. We observer that the relative increase in test accuracy using GraphMix over the baseline GNN is more pronounced when the labeled samples are fewer. This makes GraphMix particularly appealing for very few labeled data problems.

4.3. Results on Larger Datasets

We provide results on three recently proposed datasets which are relatively larger than standard benchmark datasets (Cora/Citeseer/Pubmed). We use Cora-Full dataset proposed in (Bojchevski & Günnemann, 2018) and Coauthor-CS and Coauthor-Physics datasets proposed in (Shchur et al., 2018). We took processed versions of these dataset available here [2]. We did 10 random splits of the the data into train/validation/test split. For the classes which had more than 100 samples. We choose 20 samples per class for training, 30 samples per class for validation and the remaining samples as test data. For the classes which had less than 100 samples, we chose 20% samples, per class for training, 30% samples for validation and the remaining for testing. For each split we run experiments using 100 random seeds. The statistics of these datasets is presented in Appendix Table 6 and the results are presented in Table 2. We observe that GraphMix(GCN) improves the results over GCN for all the three datasets. We note that we did minimal hyperparameter search for GraphMix(GCN) as mentioned in Appendix A.5.2.

[2]https://github.com/shchur/gnn-benchmark

Table 1. Results of node classification (% test accuracy) using 10 random Train/Validation/Test split of datasets as well as the standard split. [*] means the results are taken from the corresponding papers. We conduct 100 trials and report mean and standard deviation over the trials (refer to Table 7 in the Appendix for comparison with other methods on standard Train/Validation/Test split).

Algorithm	Cora	Citeseer	Pubmed
Random Split			
GCN	77.84±1.45	72.56±2.46	78.74±0.99
GAT	77.74±1.86	70.41±1.81	78.48±0.96
Graph U-Net	77.59±1.60	67.55±0.69	76.79±2.45
GCN (self-training)	80.41±1.78	73.62±2.11	79.81±2.85
FCN (self-training)	75.19±3.53	70.49±1.91	73.40±2.48
GraphMix (GCN)	**82.07±1.17**	**76.45±1.57**	**80.72±1.08**
GraphMix (GAT)	80.63±1.31	74.08±1.26	80.14±1.51
GraphMix (Graph-U-Net)	80.18±1.62	72.85±1.71	78.47±0.64
Standard Split			
GCN * (Kipf & Welling, 2016)	81.5	70.3	79.0
GAT * (Veličković et al., 2018)	83.0	72.5	79.0
Graph U-Net * (Gao & Ji, 2019)	**84.4**	73.2	79.6
GCN	81.30±0.66	70.61±0.22	79.86±0.34
GAT	82.70±0.21	70.40±0.35	79.05±0.64
Graph U-Net	81.74±0.54	67.69±1.10	77.73 ±0.98
GCN (self-training)	82.03±0.43	73.38±0.35	80.42±0.36
FCN (self-training)	80.30±0.75	71.50±0.80	77.40±0.37
GraphMix (GCN)	**83.94±0.57**	**74.72±0.59**	80.98±0.55
GraphMix (GAT)	83.32±0.18	73.08±0.23	**81.10±0.78**
GraphMix (Graph U-Net)	82.18±0.63	69.00 ±1.32	78.76±1.09

4.4. Semi-supervised Link Classification

In the semi-supervised Link Classification problem, the task is to predict the labels of the remaining links, given a graph and labels of a few links. Following (Taskar et al., 2004), we can formulate the link classification problem as a node classification problem. Specifically, given an original graph G, we construct a dual Graph G'. The node set V' of the dual graph corresponds to the link set E' of the original graph. The nodes in the dual graph G' are connected if their corresponding links in the graph G share a node. The attributes of a node in the dual graph are defined as the index of the nodes of the corresponding link in the

Table 2. Comparison of GraphMix with other methods (% test accuracy), for Cora-Full, Coauthor-CS, Coauthor-Physics. $*$ refers to the results reported in (Shchur et al., 2018).

Method	Cora-Full	Coauthor-CS	Coauthor-Physics
GCN*	62.2±0.6	91.1±0.5	92.8±1.0
GAT*	51.9±1.5	90.5±0.6	92.5±0.9
MoNet*	59.8±0.8	90.8±0.6	92.5±0.9
GS-Mean*	58.6±1.6	91.3±2.8	93.0±0.8
GCN	60.13±0.57	91.27±0.56	92.90±0.92
GraphMix (GCN)	**61.80±0.54**	**91.83±0.51**	**94.49±0.84**

Table 3. Results on Link Classification (%F1 score). [*] means the results are taken from the corresponding papers

Algorithm	Bit OTC	Bit Alpha
DeepWalk (Perozzi et al., 2014)	63.20	62.71
GMNN* (Qu et al., 2019)	**66.93**	**65.86**
GCN	65.72±0.38	64.00±0.19
GCN(self-training)	65.15±0.29	64.56±0.21
FCN(self-training)	60.13±0.40	59.74±0.32
GraphMix (GCN)	66.35±0.41	65.34±0.19

original graph. Using this formulation, we present results on link classification on Bit OTC and Bit Alpha benchmark datasets in the Table 3. As the numbers of the positive and negative edges are strongly imbalanced, we report the F1 score. Our results show that GraphMix(GCN) improves the performance over the baseline GCN method for both the datasets. Furthermore, the results of GraphMix(GCN) are comparable with the recently proposed state-of-the-art method GMNN (Qu et al., 2019).

4.5. Ablation Study

Since GraphMix consists of various components, some of which are common with the existing literature of semi-supervised learning, we set out to study the effect of various components by systematically removing or adding a component from GraphMix . We measure the effect of the following:

- Removing the Manifold Mixup and predicted targets from the FCN training.
- Removing the predicted targets from the FCN training.
- Removing the Manifold Mixup from the FCN training.
- Removing the Sharpening of the predicted targets.
- Removing the Average of predictions for K random perturbations of the input sample
- Using the EMA (Tarvainen & Valpola, 2017) of GNN for target prediction.

The ablation results for semi-supervised node classification are presented in Table 4. We did not do any hyperparameter tuning for the ablation study and used the best performing hyperparameters found for the results presented in Table 1. We observe that all the components of GraphMix contribute to its performance, with Manifold Mixup training of FCN contributing possibly the most. Furthermore, we observe that using the EMA model (which is an emsemble model) (Tarvainen & Valpola, 2017) for computing the predicted-targets, could improve the performance of GraphMix for all the datasets.

Table 4. Ablation study results using 10 labeled samples per class (% test accuracy). We report mean and standard deviation over ten trials.

Ablation	Cora	Citeseer	Pubmed
GraphMix	79.30±1.36	70.78±1.41	77.13±3.60
-w/o Manifold Mixup and w/o predicted-targets	68.78±3.54	61.01±1.24	72.56±1.08
-w/o predicted-targets	72.85±3.79	64.40±2.20	74.74±1.69
-w/o Manifold Mixup	69.08±5.03	62.66±1.80	74.11±0.94
-w/o Sharpening	73.25±3.41	64.65±2.21	74.97±1.47
-w/o Prediction Averaging	74.17±1.99	65.52±1.78	75.59±2.63
-with EMA	79.84±2.28	71.21±1.32	77.46±3.13

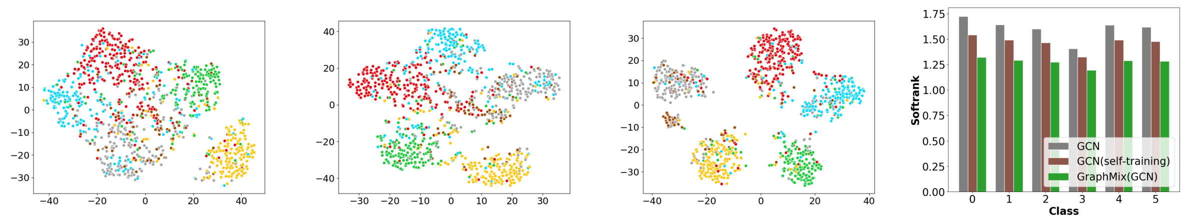
4.6. Visualization of the Learned Features

In this section, we present the analysis of the features learned by GraphMix for Cora dataset. Specifically, we present the 2D visualization of the hidden states using the t-SNE (van der Maaten & Hinton, 2008) in Figure 2a, 2b and 2c . We observe that GraphMix learns hidden states which are better separated and condensed than GCN and GCN(self-training). We further evaluate the Soft-rank (refer to Appendix A.7) of the class-specific hidden states to demonstrate that GraphMix(GCN) makes the class-specific hidden states more concentrated than GCN and GCN(self-training), as shown in Figure 2d. Refer to Appendix A.8 for 2D representation of other datasets.

5. Related Work

5.1. Semi-supervised Learning over Graph Data

There exists a long line of work for semi-supervised learning over Graph Data. Earlier work included using *Graph Laplacian Regularizer* for enforcing local smoothness over the predicted targets for the nodes (Zhu & Ghahramani, 2002; Zhu et al., 2003; Belkin et al., 2006). Another line of work learns node embedding in an unsupervised way (Perozzi et al., 2014) which can then be used as an input to any classifier, or learns the node embedding and target prediction jointly (Yang et al., 2016). Many of the recent Graph Neural Network based approaches (refer to (Zhou et al., 2018) for a review of these methods) are inspired by the success of Convolutional Neural Networks in image and text domains, defines the convolutional operators using the neighbourhood information of the nodes (Kipf & Welling, 2017; Veličković et al., 2018; Defferrard et al., 2016). These convolution operator based method exhibit state-of-the-results for semi-supervised learning over graph data, hence much of the recent attention is dedicated to proposing architectural changes to these methods (Qu et al., 2019; Gao & Ji, 2019; Ma et al., 2019). Unlike these meth-



(a) GCN (b) GCN(self-training) (c) GraphMix(GCN) (d) Class-specific Soft-Rank

Figure 2. 2D representation of the hidden states of Citeseer dataset using (a) GCN, (b) GCN(self-training), (c) GraphMix, and Soft-Rank (d).

ods, we propose a regularization technique that can be applied to any of these Graph Neural Networks which uses a parameterized transformation on the node features.

5.2. Data Augmentation

It is well known that the generalization of a learning algorithm can be improved by enlarging the training data size. Because labeling more samples is labour-intensive and costly, Data-augmentation has become *de facto* technique for enlarging the training data size, especially in the computer vision applications such as image classification. Some of the notable Data Augmentation techniques include Cutout (Devries & Taylor, 2017) and DropBlock (Ghiasi et al., 2018). In Cutout, a contiguous part of the input is zeroed out. DropBlock further extends Cutout to the hidden states. In another line of research, such as Mixup and BC-learning (Zhang et al., 2018; Tokozume et al., 2017), additional training samples are generated by interpolating the samples and their corresponding targets. Manifold Mixup (Verma et al., 2019a) proposes to augment the data in the hidden states and shows that it learns more *discriminative* features for supervised learning. Furthermore, ICT (Verma et al., 2019b) and MixMatch (Berthelot et al., 2019) extend the Mixup technique to semi-supervised learning, by computing the predicted targets for the unlabeled data and applying the Mixup on the unlabeled data and their corresponding predicted targets. Even further, for unsupervised learning, ACAI (Berthelot* et al., 2019) and AMR (Beckham et al., 2019) explore the interpolation techniques for autoencoders. ACAI interpolates the hidden states of an autoencoder and uses a critic network to constrain the reconstruction of these interpolated states to be realistic. AMR explores different ways of combining the hidden states of an autoencoder other than the convex combinations of the hidden states. Unlike, all of these techniques which have been proposed for the fixed topology datasets, in this work, we propose interpolation based data-augmentation techniques for graph-structured data.

5.3. Regularizing Graph Neural Networks

Regularizing Graph Neural Networks has drawn some attention recently. GraphSGAN (Ding et al., 2018) first uses an embedding method such as DeepWalk (Perozzi et al., 2014) and then trains generator-classifier networks in the adversarial learning setting to generate fake samples in the low-density region between sub-graphs. BVAT (Deng et al., 2019) and (Feng et al., 2019) generate adversarial perturbations to the features of the graph nodes while taking graph structure into account. While these methods improve generalization in graph-structured data, they introduce significant additional computation cost: GraphScan requires computing node embedding as a preprocessing step, BVAT and (Feng et al., 2019) require additional gradient computation for computing adversarial perturbations. Unlike these methods, GraphMix does not introduce any significant additional computation since it is based on interpolation-based techniques and self-generated targets.

6. Discussion

We presented GraphMix , a simple and efficient regularized training scheme for GNNs. GraphMix is a general scheme that can be applied to any GNN that uses a parameterized transformation on the feature vector of the graph nodes. Through extensive experiments, we demonstrated state-of-the-art performances or close to state-of-the-art performance using GraphMix on various benchmark datasets, more importantly, GraphMix improves test accuracy over vanilla GNN across all the datasets, even without doing any extensive hyperparameter search. The strong empirical results of GraphMix suggest that in parallel to designing new architectures, exploring better regularization for graph-structured data is a promising avenue for research. A future research direction is to jointly model the node features and edges of the graph, which can be further used for generating the synthetic interpolated nodes and their corresponding connectivity to the other nodes in the graph. This will alleviate the need to train the auxiliary FCN in GraphMix .

References

Beckham, C., Honari, S., Verma, V., Lamb, A., Ghadiri, F., Devon Hjelm, R., Bengio, Y., and Pal, C. On Adversarial Mixup Resynthesis. *arXiv e-prints*, art. arXiv:1903.02709, Mar 2019.

Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, December 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248632.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. MixMatch: A Holistic Approach to Semi-Supervised Learning. *arXiv e-prints*, art. arXiv:1905.02249, May 2019.

Berthelot*, D., Raffel*, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1fQSiCcYm.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pp. 92–100, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0. doi: 10.1145/279943.279962. URL http://doi.acm.org/10.1145/279943.279962.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

Chapelle, O., Schlkopf, B., and Zien, A. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3844–3852. 2016.

Deng, Z., Dong, Y., and Zhu, J. Batch virtual adversarial training for graph convolutional networks. *CoRR*, abs/1902.09192, 2019. URL http://arxiv.org/abs/1902.09192.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL http://arxiv.org/abs/1708.04552.

Ding, M., Tang, J., and Zhang, J. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pp. 913–922, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6014-2. doi: 10.1145/3269206.3271768. URL http://doi.acm.org/10.1145/3269206.3271768.

Feng, F., He, X., Tang, J., and Chua, T. Graph adversarial training: Dynamically regularizing based on graph structure. *CoRR*, abs/1902.08226, 2019. URL http://arxiv.org/abs/1902.08226.

Gao, H. and Ji, S. Graph u-nets. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2083–2092, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/gao19a.html.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Dropblock: A regularization method for convolutional networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10727–10737. Curran Associates, Inc., 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. In Saul, L. K., Weiss, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 17*, pp. 529–536. 2005.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017.

Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data. *ArXiv*, abs/1506.05163, 2015.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklr3j0cKX.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. Audio augmentation for speech recognition. In *INTERSPEECH*, 2015.

Kumar, S., Spezzano, F., Subrahmanian, V., and Faloutsos, C. Edge weight prediction in weighted signed networks. In *ICDM*, 2016.

Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. Rev2: Fraudulent user prediction in rating platforms. In *WSDM*, 2018.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *CoRR*, abs/1610.02242, 2016. URL http://arxiv.org/abs/1610.02242.

Lee, D.-H. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. 2013.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Lu, Q. and Getoor, L. Link-based classification. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 496–503. AAAI Press, 2003. ISBN 1-57735-189-4. URL http://dl.acm.org/citation.cfm?id=3041838.3041901.

Ma, J., Cui, P., Kuang, K., Wang, X., and Zhu, W. Disentangled graph convolutional networks. In *ICML*, 2019.

Miyato, T., ichi Maeda, S., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016. URL http://arxiv.org/abs/1611.08402.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *arXiv e-prints*, art. arXiv:1904.08779, Apr 2019.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD*, 2014.

Qu, M., Bengio, Y., and Tang, J. GMNN: Graph Markov neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5241–5250, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, January 2009. ISSN 1045-9227. doi: 10.1109/TNN.2008.2005605. URL http://dx.doi.org/10.1109/TNN.2008.2005605.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018. URL http://arxiv.org/abs/1811.05868.

Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lfF2NYvH.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems 30*, pp. 1195–1204, 2017.

Taskar, B., Wong, M.-F., Abbeel, P., and Koller, D. Link prediction in relational data. In *NIPS*, 2004.

Tokozume, Y., Ushiku, Y., and Harada, T. Between-class learning for image classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5486–5494, 2017.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9: 2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *ICLR*, 2019.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the*

36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, pp. 6438–6447, Long Beach, California, USA, 09–15 Jun 2019a. PMLR. URL http://proceedings.mlr.press/v97/verma19a.html.

Verma, V., Lamb, A., Juho, K., Bengio, Y., and Lopez-Paz, D. Interpolation consistency training for semi-supervised learning. In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, 2019b. doi: 10.24963/ijcai.2019. URL https://doi.org/10.24963/ijcai.2019.

Weston, J., Ratle, F., Mobahi, H., and Collobert, R. Deep learning via semi-supervised embedding. In Montavon, G., Orr, G., and Müller, K. R. (eds.), *In Neural Networks: Tricks of the Trade*. Springer, second edition, 2012.

Xie, Z., Wang, S. I., Li, J., Lévy, D., Nie, A., Jurafsky, D., and Ng, A. Y. Data noising as smoothing in neural network language models. *ArXiv*, abs/1703.02573, 2017.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/xu18c.html.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., and Sun, M. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL http://arxiv.org/abs/1812.08434.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. Technical report, 2002.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

Algorithm 1 GraphMix : A procedure for improved training of Graph Neural Networks (GNN)

```
1:  Input: A GCN: $G_\theta(X, A)$, a FCN: $F_\theta(X, \alpha)$ which shares parameters with the GCN. Beta distribution parameter $\alpha$ for Manifold Mixup . Number of random perturbations $K$, Sharpening temperature $T$. maximum value of weight $\gamma$ in the weighted averaging of supervised FCN loss and unsupervised FCN loss. Number of epochs $N$. $w(t)$: rampup function for increasing the importance unsupervised loss in FCN training. $(X_L, Y_L)$ represents labeled samples and $X_U$ represents unlabeled samples.
2:  for $t = 1$ to $N$ do
3:    i = random(0,1)  // generate randomly 0 or 1
4:    if i=0 then
5:      $\mathcal{L}_{\text{supervised}} = \mathcal{L}_{\text{MM}}((\mathbf{X}_l, \mathbf{Y}_l), F_\theta, \alpha)$  // supervised loss from FCN using the Manifold Mixup
6:      for $k = 1$ to $K$ do
7:        $\hat{X}_{U,k} = RandomPerturbations(X_U)$  // Apply $k^{th}$ round of random perturbation to $X_U$
8:      end for
9:      $\bar{Y}_U = \frac{1}{K}\sum_k g(Y \mid \hat{X}_{U,k}; \theta, A)$  // Compute average predictions across K perturbations of $X_U$ using the GCN
10:     $Y_U = \text{Sharpen}(\bar{Y}_U, T)$  // Apply temperature sharpening to the average prediction
11:     $\mathcal{L}_{\text{unsupervised}} = \mathcal{L}_{\text{MM}}((\mathbf{X}_u, \hat{\mathbf{Y}}_u), F_\theta, \alpha)$  // unsupervised loss from FCN using the Manifold Mixup
12:     $\mathcal{L} = \mathcal{L}_{\text{supervised}} + w(t) * \mathcal{L}_{\text{unsupervised}}$  // Total loss is the weighted sum of supervised and unsupervised FCN loss
13:   else
14:     $\mathcal{L} = \mathcal{L}(G_\theta(\mathbf{X}_l), \mathbf{Y}_l)$  // Loss using the vanilla GCN
15:   end if
16:   Minimize $\mathcal{L}$ using gradient descent based optimizer such as SGD.
17: end for
```

A. Appendix

A.1. Algorithm

The procedure for GraphMix training is given in Algorithm 1.

A.2. Datasets

The statistics of standard benchmark datasets as well as the number of training/validation/test nodes is presented in Table 5. The statistics of larger datasets in given in Table 6.

A.3. Comparison with State-of-the-art Methods

We present the comparion of GraphMix with the recent state-of-the-art methods as well as earlier methods using the standard Train/Validation/Test split in Table 7.

A.4. Results with fewer labeled samples

We further evaluate the effectiveness of GraphMix in the learning regimes where fewer labeled samples exist. For each class, we randomly sampled $K \in \{5, 10\}$ samples for training and the same number of samples for the validation. We used all the remaining labeled samples as the test set. We repeated this process for 10 times. The results in Table 8 show that GraphMix achieves even better improvements when the labeled samples are fewer (Refer to Table 1 for results with 20 training samples per class).

A.5. Implementation and Hyperparameter Details

We use the standard benchmark architecture as used in GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018) and GMNN (Qu et al., 2019), among others. This architecture has one hidden layer and the graph convolution is applied twice : on the input layer and on the output of the hidden layer. The FCN in GraphMix shares the parameters with the GCN.

GraphMix introduces four additional hyperparameters, namely the α parameter of Beta distribution used in Manifold Mixup training of the FCN, the maximum weighing coefficient γ which controls the trade-off between the supervised loss and the unsupervised loss (loss computed using the predicted-targets) of FCN, the temparature T in sharpening and the number of random perturbations K applied to the input data for the averaging of the predictions.

We conducted minimal hyperparameter seach over only α and γ and fixed the hyperparameters T and K to 0.1 and 10 respectively. The other hyperparameters were set to the best values for underlying GNN (GCN or GAT), including the learning rate, the $L2$ decay rate, number of units in the hidden layer etc. We observed that GraphMix is not very sensitive to the values of α and γ and similar values of these hyperparameters work well across all the benchmark datasets. Refer to Appendix A.5 and A.6 for the details about the hyperparameter values and the procedure used for the best hyperparameters selection.

A.5.1. For results reported in Section 4.2

For GCN and GraphMix(GCN), we used Adam optimizer with learning rate 0.01 and $L2$-decay 5e-4, the number of units in the hidden layer 16 , dropout rate in the input layer and hidden layer was set to 0.5 and 0.0, respectively. For GAT and GraphMix(GAT), we used Adam optimizer with learning rate 0.005 and $L2$-decay 5e-4, the number of units in the hidden layer 8 , and the dropout rate in the input layer and hidden layer was searched from the values $\{0.2, 0.5, 0.8\}$.

For α and γ of GraphMix(GCN) and GraphMix(GAT) , we searched over the values in the set $[0.0, 0.1, 1.0, 2.0]$ and $[0.1, 1.0, 10.0, 20.0]$ respectively.

For GraphMix(GCN) : $\alpha = 1.0$ works best across all the datasets. $\gamma = 1.0$ works best for Cora and Citeseer and

Table 5. Dataset statistics.

Dataset	# Nodes	# Edges	# Features	# Classes	# Training	# Validation	# Test
Cora	2,708	5,429	1,433	7	140	500	1,000
Citeseer	3,327	4,732	3,703	6	120	500	1,000
Pubmed	19,717	44,338	500	3	60	500	1,000
Bitcoin Alpha	3,783	24,186	3,783	2	100	500	3,221
Bitcoin OTC	5,881	35,592	5,881	2	100	500	5,947

Table 6. Dataset statistics for Larger datasets

Datasets	Classes	Features	Nodes	Edges
Cora-Full	67	8710	18703	62421
Coauthor-CS	15	6805	18333	81894
Coauthor-Physics	5	8415	34493	247962
NELL	210	5414	65755	266144

Table 7. Comparison of GraphMix with other methods (% test accuracy), for Cora, Citeseer and Pubmed.

Method	Cora	Citeseer	Pubmed
Results reported from the literature			
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	79.0%
MoNet (Monti et al., 2016)	81.7 $\pm$ 0.5%	—	78.8 $\pm$ 0.3%
GAT (Veličković et al., 2018)	83.0 $\pm$ 0.7%	72.5 $\pm$ 0.7%	79.0 $\pm$ 0.3%
GraphScan (Ding et al., 2018)	83.3 $\pm$1.3	73.1$\pm$1.8	—
DisenGCN (Ma et al., 2019)	83.7%	73.4%	80.5%
Graph U-Net (Gao & Ji, 2019)	**84.4%**	73.2%	79.6%
BVAT (Deng et al., 2019)	83.6$\pm$0.5	**74.0$\pm$0.6**	79.9$\pm$0.4
Our Experiments			
GCN	81.30$\pm$0.66	70.61$\pm$0.22	79.86$\pm$0.34
GAT	82.70$\pm$0.21	70.40$\pm$0.35	79.05$\pm$0.64
Graph U-Net	81.74$\pm$0.54	67.69$\pm$1.10	77.73 $\pm$0.98
GCN (self-training)	82.03$\pm$0.43	73.38$\pm$0.35	80.42$\pm$0.36
FCN (self-training)	80.30$\pm$0.75	71.50$\pm$0.80	77.40$\pm$0.37
GraphMix (GCN)	**83.94$\pm$0.57**	**74.52$\pm$0.59**	80.98$\pm$0.55
GraphMix (GAT)	83.32$\pm$0.18	73.08$\pm$0.23	**81.10$\pm$0.78**
GraphMix (Graph U-Net)	82.18$\pm$0.63	69.00 $\pm$1.32	78.76$\pm$1.09

$\gamma = 10.0$ works best for Pubmed.

For GraphMix(GAT) : $\alpha = 1.0$ works best for Cora and Cite-

Table 8. Results using less labeled samples (% test accuracy). K referes to the number of labeled samples per class.

Algorithm	Cora		Citeseer		Pubmed	
	$K=5$	$K=10$	$K=5$	$K=10$	$K=5$	$K=10$
GCN	66.39±4.26	72.91±3.10	55.61±5.75	64.19±3.89	66.06±3.85	75.57±1.58
GAT	68.17±5.54	73.88±4.35	55.54±1.82	61.63±0.42	64.24±4.79	73.60±1.85
Graph U-Net	64.42±5.44	71.48±3.03	49.43±5.81	61.16±3.47	65.05±4.69	68.65±3.69
GraphMix (GCN)	**71.99±6.46**	**79.30±1.36**	**58.55±2.26**	**70.78±1.41**	**67.66±3.90**	**77.13±3.60**
GraphMix (GAT)	72.01±6.68	75.82±2.73	57.6±0.64	62.24±2.90	66.61±3.69	75.96±1.70
GraphMix (Graph U-Net)	66.84±6 5.10	73.14±3.17	54.39±5.07	64.36±3.48	67.40±5.33	70.43±3.75

seer and $\alpha = 0.1$ works best for Pubmed. $\gamma = 1.0$ works best for Cora and Citeseer and $\gamma = 10.0$ works best for Pubmed. Input droputrate=0.5 and hidden dropout rate=0.5 work best for Cora and Citeseer and Input dropout rate=0.2 and hidden dropout rate =0.2 work best for Pubmed.

We conducted all the experiments for 2000 epochs. The value of weighing coefficient $w(t)$ in Algorithm 1) is increased from 0 to its maximum value γ from epoch 500 to 1000 using the sigmoid ramp-up of Mean-Teacher (Tarvainen & Valpola, 2017).

A.5.2. Hyperparameter Details for Results in Section 4.3

For all the experiments we use the standard architecture mentioned in Section A.5 and used Adam optimizer with learning rate 0.001 and 64 hidden units in the hidden layer. For Coauthor-CS and Coauthor-Physics, we trained the network for 2000 epochs. For Cora-Full, we trained the network for 5000 epochs because we observed the training loss of Cora-Full dataset takes longer to converge.

For Coauthor-CS and Coauthor-Physics: We set the input layer dropout rate to 0.5 and weight-decay to 0.0005, both for GCN and GraphMix(GCN) . We did not conduct any hyperparameter search over the GraphMix hyperparameters α, λ_{max}, temparature T and number of random permutations K applied to the input data for GraphMix(GCN) for these two datasets, and set these values to 1.0, 1.0, 0.1 and 10 respectively.

For Cora-Full dataset: We found input layer dropout rate 0.2 and weight-decay 0.0 to be the best for both GCN and GraphMix(GCN) . For GraphMix(GCN) we fixed α, temparature T and number of random permutations K to 1.0 0.1 and 10 respectively. For λ_{max}, we did search over $\{1.0, 10.0, 20.0\}$ and found that 10.0 works best.

For all the GraphMix(GCN) experiments, the value of weighing coefficient $w(t)$ in Algorithm 1) is increased from 0 to its maximum value γ_{max} from epoch 500 to 1000 using the sigmoid ramp-up of Mean-Teacher (Tarvainen & Valpola, 2017).

A.5.3. For results reported in Section 4.4

For α of GraphMix(GCN) , we searched over the values in the set $[0.0, 0.1, 0.5, 1.0]$ and found that 0.1 works best for both the datasets. For γ, we searched over the values in the set $[0.1, 1.0, 10.0]$ and found that 0.1 works best for both the datasets. We conducted all the experiments for 150 epochs. The value of weighting coefficient $w(t)$ in Algorithm 1) is increased from 0 to its maximum value γ from epoch 75 to 125 using the sigmoid ramp-up of Mean-Teacher (Tarvainen & Valpola, 2017).

Both for GraphMix(GCN) and GCN, we use Adam optimizer with learning rate 0.01 and $L2$-decay 0.0, the number of units in the hidden layer 128 , dropout rate in the input layer was set to 0.5.

A.5.4. For results reported in Section A.4

For α of GraphMix(GCN) , we searched over the values in the set $[0.0, 0.1, 0.5, 1.0]$ and found that 0.1 works best across all the datasets. For γ, we searched over the values in the set $[0.1, 1.0, 10.0]$ and found that 0.1 and 1.0 works best across all the datasets. Rest of the details for GraphMix(GCN) and GCN are same as Section A.5.1.

A.6. Hyperparameter Selection

For each configuration of hyperparameters, we run the experiments with 100 random seeds. We select the hyperparameter configuration which has the best validation accuracy averaged over these 100 trials. With this best hyperparameter configuration, for 100 random seeds, we train the model again and use the validataion set for model selection (i.e. we report the test accuracy at the epoch which has best validation accuracy.)

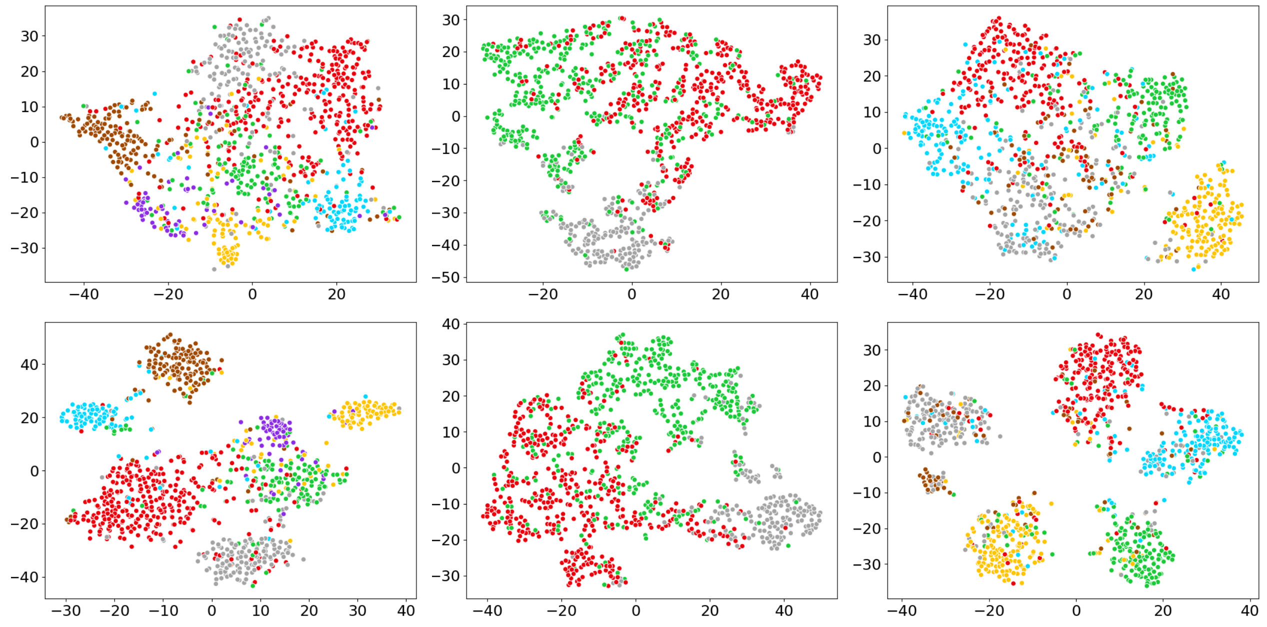


Figure 3. T-SNE of hidden states for Cora (left), Pubmed (middle), and Citeseer (right). Top row is GCN baseline, bottom row is GraphMix.

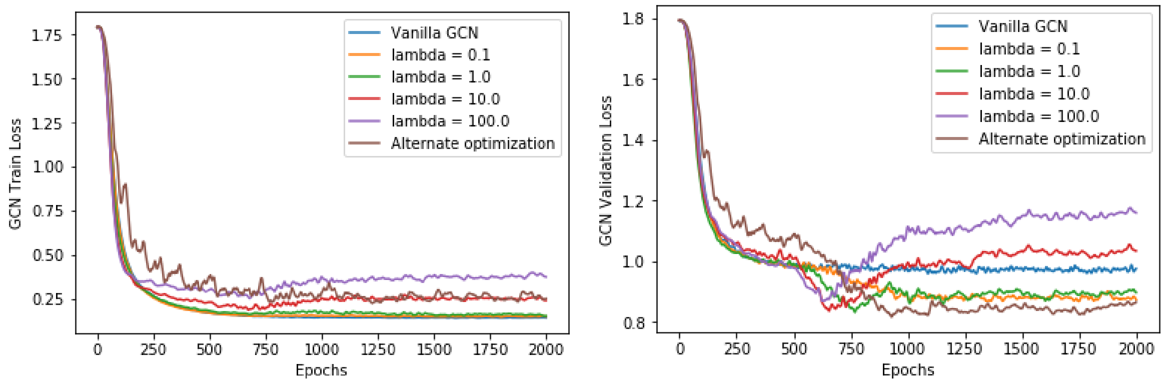


Figure 4. GCN train loss and validation loss for Alternate optimization vs. weighted joint optimization. lambda = X.X represents the value of λ for simultaneous optimization in Eq 7.

A.7. Soft-Rank

Let **H** be a matrix containing the hidden states of all the samples from a particular class. The Soft-Rank of matrix **H** is defined by the sum of the singular values of the matrix divided by the largest singular value. A lower Soft-Rank implies fewer dimensions with substantial variability and it provides a continuous analogue to the notion of rank from matrix algebra. This provides evidence that the concentration of class-specific states observed when using GraphMix in Figure 3 can be measured directly from the hidden states and is not an artifact of the T-SNE visualization.

A.8. Feature Visualization

We present the 2D visualization of the hidden states learned using GCN and GraphMix(GCN) for Cora, Pubmed and Citeseer datasets in Figure 3. We observe that for Cora and Citeseer, GraphMix learns substantially better hidden states than GCN. For Pubmed, we observe that although there is no clear separation between classes, "Green" and "Red" classes overlap less using the GraphMix, resulting in better hidden states.

A.9. Joint Optimization vs Alternate optimization

In this Section, we discuss the effect of hyperparameter λ, that is used to compute the weighted sum of GCN and FCN losses in in Eq 7. In Figure 4, we see that a wide range of λ (from 0.1 to 1.0) achieves better validation accuracy than the vanilla GCN. Furthermore, alternate optimization of the FCN loss and the GCN loss achieves substantially better validation accuracy than the simultaneous optimization.