GraphMix : Regularized 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]
[1]Aalto University, Finland
[2]Mila - Québec Artificial Intelligence Institute, Montréal, Canada
[3]Université de Montréal, Canada
[4] HEC, Montréal, Canada

Abstract

We present *GraphMix*, a regularization technique 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: the *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

Early work for learning from Graph structured data includes (Gori et al.; Scarselli et al., 2009) which propose a neural network that can directly process most type 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 important problem of Semi-supervised learning over graph structured data (Zhou et al., 2018). However, much of this research effort has been dedicated to developing novel architectures.

Unlike many existing works which try to come up with the new architectures, we focus on architecture-agnostic regularization techniques for graph neural networks based semi-supervised object classification. Data Augmentation based regularization has been shown to be very effective in other types of neural networks but how to apply these techniques in graph neural networks is still under-explored. Our proposed method GraphMix [2] [3]is inspired by interpolation based data augmentation techniques (Zhang et al., 2018; Verma et al., 2019a) but is changed appropriately to make it suitable for graph structured data. Furthermore, GraphMix also utilizes the self-target-prediction (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Verma et al., 2019b; Berthelot et al., 2019) based data-augmentation. We show that with our proposed regularization techniques, we can achieve state-of-the-art performance

[*]Correspondence email: vikasverma.iitm@gmail.com, vikas.verma@aalto.fi
[2]code available at https://github.com/vikasverma1077/GraphMix
[3]Full version of paper availble at GraphMix

Workshop on Graph Representation Learning @ 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

even when using simpler graph neural network architectures such as Graph Convolutional Networks (Kipf & Welling, 2017) and without incurring any significant additional computation cost.

2 Problem Definition and Preliminaries

2.1 Problem Setup

We are interested in the problem of semi-supervised object classification using graph structured data. We can formally define such graph structured data as $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V}$ represents the set of nodes $\{v_1, \ldots, v_n\}$, and E is the set of edges between the nodes of $\mathcal{V}$.

Each node v_i in the graph has a corresponding d-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^d$. The feature vectors of all the nodes $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ are stacked together to form the entire feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Each node belongs to one out of C classes and can be labeled with a C-dimensional one-hot vector $\mathbf{y}_i \in \{0, 1\}^C$. Given the labels of $\mathbf{Y}_L$ for few of the labeled nodes $\mathcal{V}_L \subset \mathcal{V}$, the task is to predict the the labels $\mathbf{Y}_U$ of the remaining nodes $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_L$.

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. This is done by using an aggregation function that takes as an input the representations of all the samples and 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:

$$\mathbf{H}_{l+1} = a(\mathbf{H}_l \mathbf{W}, E) \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{k \times k'}$ is a linear transformation matrix, k' is the dimension of $(l+1)_{th}$ layer and a is the aggregation function that utilizes the graph structure.

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. In the context of supervised leaning, Mixup (Zhang et al., 2018) trains a neural network on the convex combination of input and targets, where as 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 a constraint 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) class-conditioned hidden states. Furthermore, in the context of semi-supervised learning, ICT (Verma et al., 2019b) and MixMatch (Berthelot et al., 2019), extend the Mixup technique by computing the predicted targets for the unlabeled data and applying the Mixup on the unlabeled data and their corresponding predicted targets.

3 GraphMix

GraphMix augments the vanilla GNN with a Fully Connected Network (FCN) via parameter sharing. The FCN loss is computed using the *Manifold Mixup* and the GNN loss is computed in the standard way. Both of these losses are optimized in an alternating fashion during training. *Manifold Mixup* has been shown to learn better features. The use of *Manifold Mixup* for FCN training facilitates learning better features, which are used in the GNN training via parameter sharing. The predicted targets from the GNN are used to augment the training set of the FCN. In this way, both FCN and GNN facilitate each other's learning process. At inference time, the predictions are made using only GNN. The diagrammatic representation of GraphMix is presented in Figure 1 and the full algorithm is presented in Algorithm 1.

Some implementation considerations. 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 hidden layer. Applying *mixup* only in the input layer also resulted in underfitting and did not improve test accuracy.

The GraphMix framework 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) and GAT (Veličković et al., 2018) using the GraphMix , however, this framework can also be applied to more recent GNNs such as Graph U-Net (Gao & Ji, 2019) and DisenGCN (Ma et al., 2019), which may facilitate in improving the state-of-the-art even further.

The performance of self-supervision based algorithms such as GraphMix is greatly affected by the accuracy of the predicted targets. To improve the accuracy of the predicted targets, we applied the average of the model prediction on K random perturbations of an input sample as discussed in Section 3.1 and sharpening as described in Section 3.2. Further, we draw similarities and difference of GraphMix w.r.t. Co-training framework in the Section 6.4.

The diagrammatic representation of GraphMix is presented in Figure 1 and the full algorithm is presented in Algorithm 1.

3.1 Accurate Target Prediction for Unlabeled data

Recent state-of-the-art semi-supervised learning methods use a *teacher* model to make accurate target predictions on unlabeled data. These predicted targets on the unlabeled data are used as "true labels" for further training of the model. 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). Another recently proposed method for accurate target predictions for unlabeled data is to use the average of the predicted targets across K random augmentations of the input sample (Berthelot et al., 2019). Along these lines, in this work, we compute the predicted target as the average of predictions on K drop-out versions of the input sample. We also used the EMA of the student model but it did not improve test accuracy across all the datasets (see Section 6.3 for details).

3.2 Entropy Minimization

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). In this work, we use Sharpening for entropy minimization. The Sharpening function over the model prediction $p(y|x, \theta)$ can be formally defined as follows (Berthelot et al., 2019), where T is the temperature hyperparameter and C is the number of classes:

$$\text{Sharpen}(p_i, T) := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{C} p_j^{\frac{1}{T}} \tag{2}$$

4 Experiments

4.1 Results

In this section, we present the results of GraphMix and compare it against several baselines. For baselines, we choose the vanilla GCN, and the recent state-of-the art methods GAT (Veličković et al., 2018) and GMNN (Qu et al., 2019). To underline the importance of the shared parameters between FCN and GCN in GraphMix , we used two additional baselines: in the first one, we trained the GCN with self-generated predicted targets, and in the second one, we trained the FCN with self-generated predicted targets, named "GCN(with predicted-targets)" and "FCN(with predicted-targets)" respectively in Table 1.

We observe that for Cora and Citeseer, GraphMix achieved significant improvements over the strong baselines such as GCN, GAT and GMNN. For Pubmed, GraphMix improved upon GCN and GAT

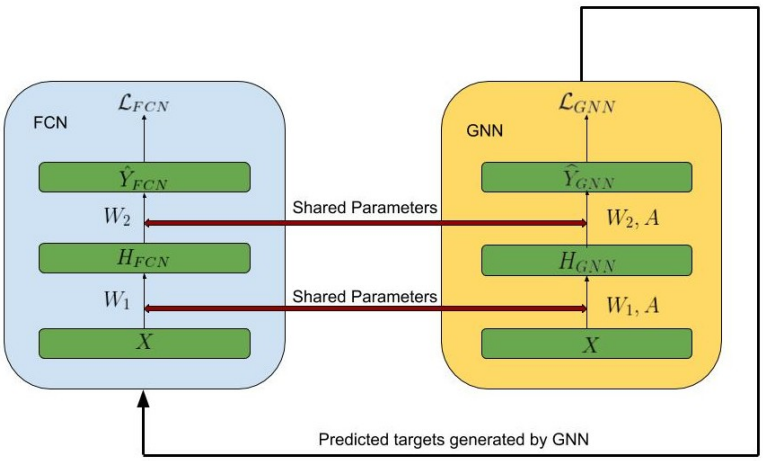


Figure 1: The procedure for training with GraphMix . The Fully-Connected Network (FCN) and the Graph Neural Network (GNN) share parameters. The FCN is trained using Manifold Mixup by interpolating the hidden states H_{FCN} and the corresponding labels Y. This leads to better features in the GNN as a result of the parameter sharing. The targets predicted 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.

Table 1: Results of object classification (%). [*] means the results are taken from the corresponding papers. We conduct 10 trials and report mean and standard deviation over the trials.

Algorithm	**Cora**	**Citeseer**	**Pubmed**
GCN * (Kipf & Welling, 2016)	81.5	70.3	79.0
GAT * (Veličković et al., 2018)	83.0	72.5	79.0
GMNN * (Qu et al., 2019)	83.7	73.1	81.8
GCN	81.3 ± 0.66	70.61 ± 0.22	79.86 ± 0.34
GCN (with predicted-targets)	82.03±0.43	73.38± 0.35	**82.42± 0.36**
FCN (with predicted-targets)	80.30±0.75	71.50± 0.80	77.40± 0.37
GraphMix	**83.94± 0.57**	**74.52±0.59**	80.98± 0.55

but was worse than GMNN. More interestingly, we obtained best results for Pubmed by just using the GCN(with predicted targets). We further present results on larger datasets (Cora-Full, Co-author-CS and Co-author-Physics) in Appendix Section 6.8 respectively.

5 Discussion

We presented GraphMix , a simple and efficient regularization technique for the graph neural networks. GraphMix is a general technique that can be applied to any graph neural network 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 this simple regularization technique on various benchmark datasets, more importantly, GraphMix improves test accuracy over vanilla GNN across all the datasets, even without doing any extensive hyperparameter search. Further, we conduct a systematic ablation study to understand the effect of different components in the performance of GraphMix . This suggests that in parallel to designing new architectures, exploring better regularization for graph structured data is a promising avenue for research.

Acknowledgments

Authors thank Petar Veličković (Google DeepMind) and David Lopez-Paz (Facebook AI Research) for helpful discussions and comments. Authors also thank Compute Canada for providing computational resources used in this work.

References

Berthelot, David, Carlini, Nicholas, Goodfellow, Ian, Papernot, Nicolas, Oliver, Avital, and Raffel, Colin. MixMatch: A Holistic Approach to Semi-Supervised Learning. *arXiv e-prints*, art. arXiv:1905.02249, May 2019.

Blum, Avrim and Mitchell, Tom. 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, Aleksandar and Günnemann, Stephan. 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, Joan, Zaremba, Wojciech, Szlam, Arthur, and LeCun, Yann. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

Defferrard, Michaël, Bresson, Xavier, and Vandergheynst, Pierre. 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.

Gao, Hongyang and Ji, Shuiwang. Graph u-nets. In Chaudhuri, Kamalika and Salakhutdinov, Ruslan (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.

Gilmer, Justin, Schoenholz, Samuel S, Riley, Patrick F, Vinyals, Oriol, and Dahl, George E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Gori, Marco, Monfardini, Gabriele, and Franco, Scarselli. A new model for learning in graph domains. *IEEE International Joint Conference on Neural Networks*.

Hamilton, Will, Ying, Zhitao, and Leskovec, Jure. Inductive representation learning on large graphs. In *NIPS*, 2017.

Henaff, Mikael, Bruna, Joan, and LeCun, Yann. Deep convolutional networks on graph-structured data. *ArXiv*, abs/1506.05163, 2015.

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

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

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

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

Ma, Jianxin, Cui, Peng, Kuang, Kun, Wang, Xin, and Zhu, Wenwu. Disentangled graph convolutional networks. In *ICML*, 2019.

Qu, Meng, Bengio, Yoshua, and Tang, Jian. GMNN: Graph Markov neural networks. In Chaudhuri, Kamalika and Salakhutdinov, Ruslan (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, Franco, Gori, Marco, Tsoi, Ah Chung, Hagenbuchner, Markus, and Monfardini, Gabriele. 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, Oleksandr, Mumme, Maximilian, Bojchevski, Aleksandar, and Günnemann, Stephan. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018. URL http://arxiv.org/abs/1811.05868.

Tarvainen, Antti and Valpola, Harri. 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.

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

Veličković, Petar, Cucurull, Guillem, Casanova, Arantxa, Romero, Adriana, Liò, Pietro, and Bengio, Yoshua. Graph attention networks. In *ICLR*, 2018.

Veličković, Petar, Fedus, William, Hamilton, William L, Liò, Pietro, Bengio, Yoshua, and Hjelm, R Devon. Deep graph infomax. In *ICLR*, 2019.

Verma, Vikas, Lamb, Alex, Beckham, Christopher, Najafi, Amir, Mitliagkas, Ioannis, Lopez-Paz, David, and Bengio, Yoshua. Manifold mixup: Better representations by interpolating hidden states. In Chaudhuri, Kamalika and Salakhutdinov, Ruslan (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, Vikas, Lamb, Alex, Juho, Kannala, Bengio, Yoshua, and Lopez-Paz, David. Interpolation consistency training for semi-supervised learning. In Kraus, Sarit (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.

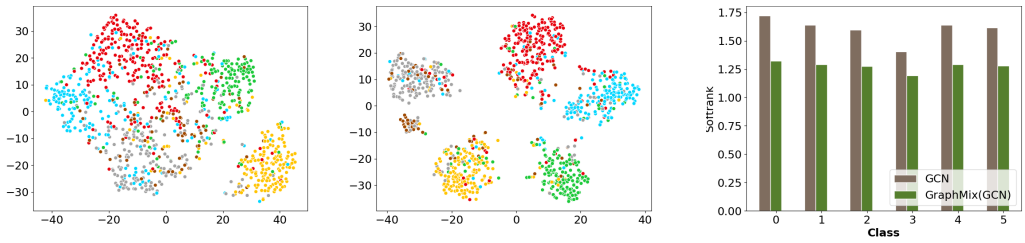
Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N., and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Zhou, Jie, Cui, Ganqu, Zhang, Zhengyan, Yang, Cheng, Liu, Zhiyuan, and Sun, Maosong. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL http://arxiv.org/abs/1812.08434.

6 Appendix

6.1 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 and 2b. We observe that GraphMix learns hidden states which are better separated and condensed. We further evaluate the Soft-rank (refer to Appendix 6.2) of the class-specific hidden states to demonstrate that GraphMix(GCN) makes the class-specific hidden states more concentrated as shown in Figure 2c.



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

Figure 2: 2D representation of the hidden states of Cora dataset using (a)GCN and (b)GraphMix, and Soft-Rank of Class-specific hidden states (lower Soft-Rank reflects more concentrated class-specific hidden states)

6.2 Soft-Rank

Let $\mathbf{H}$ be a matrix containing the hidden states of all the samples from a particular class. The Soft-Rank of matrix $\mathbf{H}$ 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 2 can be measured directly from the hidden states and is not an artifact of the T-SNE visualization.

6.3 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 results are presented in Table 2. We did not do any hyperparameter tuning for the ablation study and used the best performing hyperparameters found for the results presented in the Table 1. We observe that all the components of GraphMix contribute to its performance, except for the Pubmed dataset, for which not using the Manifold Mixup in the FCN training results in slightly better performance. We also observe that using the EMA model (Tarvainen & Valpola, 2017) for pseudolabel computation results in improved performance for Citeseer but decreased performance for Pubmed. It can be the effect of not doing the hyperparameter search when adding the EMA to the GraphMix . We leave this exploration for future work.

Ablation	Cora	Citeseer	Pubmed
GraphMix	**83.94±0.57**	**74.52±0.59**	80.98±0.55
-without Manifold Mixup, without predicted targets	79.98±0.27	70.80±0.46	79.05±0.26
-without predicted-targets	81.86±0.41	71.30±0.14	79.66±0.14
-without Manifold Mixup	83.57±0.79	73.96±0.76	**81.08±0.45**
-no Sharpening	80.20±0.23	71.30±0.27	80.06±0.18
-no Averaging of predictions	83.32±0.27	73.47±0.33	80.52±0.59
-with EMA	83.82±0.76	74.92±0.57	80.38±0.59

Table 2: Ablation study results. We report mean and standard deviation over ten trials.

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

Input: A GCN: $g(X, A, \theta)$, a FCN: $f(X, \theta, \lambda)$ which shares parameters with the GCN. Beta distribution parameter α for *Manifold Mixup* . Number of random perturbations K, Sharpening temperature T. Consistency parameter γ. Number of epochs N. $\gamma(t)$: rampup function for increasing the importance of consistency regularization.
for $t = 1$ **to** N **do**
  i = random(0,1) *// generate randomly 0 or 1*
  if i=0 **then**
    $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$ *// Sample a mixing coefficient from Beta distribution*
    $\mathcal{L}_{sup} = \mathcal{L}\big(f(X_L, \theta, \lambda), Y_L\big)$ *// supervised loss from FCN using the Manifold Mixup*
    for $k = 1$ **to** K **do**
      $\hat{X}_{U,k} = RandomPerturbations(X_U)$ *// Apply k^{th} round of random perturbation to X_U*
    end for
    $\bar{Y}_U = \frac{1}{K} \sum_k g(Y \mid \hat{X}_{U,k}; \theta, A)$ *// Compute average predictions across K perturbations of X_U using the GCN*
    $Y_U = \mathrm{Sharpen}(\bar{Y}_U, T)$ *// Apply temperature sharpening to the average prediction*
    $\mathcal{L}_{usup} = \mathcal{L}\big(f(X_U, \theta, \lambda), Y_U\big)$ *// unsupervised loss from FCN using the Manifold Mixup*
    $\mathcal{L} = \mathcal{L}_{sup} + \gamma(t) * \mathcal{L}_{usup}$ *// Total loss is the weighted sum of supervised and unsupervised FCN loss*
  else
    $\mathcal{L} = \mathcal{L}\big(g(X_L, \theta, A), Y_L\big)$ *// Loss using the vanilla GCN*
  end if
end for
return $\mathcal{L}$

6.4 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 if we had enough 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 FCN. The 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 results in reduced accuracy. This is due to the fact that the number of labeled samples for training the FCN are sufficiently low and hence the FCN does not make accurate enough predictions. Another important difference is that unlike the co-training framework, FCN and GNN do not use completely distinct views of the data: the FCN uses feature vectors $\mathbf{X}$ and the GNN uses the feature vector and edge connectivity $(\mathbf{X}, E)$.

6.5 Algorithm

The algorithmic version of GraphMix is presented in Algorithm 1.

6.6 Datasets

We use three standard benchmark citation network datasets for semi-supervised with Graph Neural Networks, 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 access has access to the class labels of only few of the nodes. The statistics of these datasets as well as the number of training/validation/test nodes is presented in Table 3.

Table 3: 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

6.7 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 max-consistency coefficient γ_{max} which controls the trade-off between the supervised loss and the unsupervised loss (loss computed using the pseudolables) of FCN, the temparature T in sharpening and the number of random perturbations K applied to the input data for averaging of the predictions.

We conducted minimal hyperparameter seach over only α and γ_{max} and fixed the hyperparameters T and K to 0.1 and 10 respectively. The other hyperparameters were set to the best values for GCN, 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_{max} and similar values of these hyperparameters work well across all the benchmark datasets.

For α, we searched over the values in the set $[0.1, 1.0, 2.0]$ and found that 1.0 works best across all the datasets. For γ_{max}, we searched over the values in the set $[1.0, 10.0, 20.0]$ and found that 10.0 works best for Cora and Citeseer and 1.0 works best for Pubmed. We conducted all the experiments for 2000 epochs. The value of consistency coefficient γ (line 13 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). We used Adam optimizer with learning rate 0.01 and $L2$-decay 5e-4. The number of units in the hidden layer was 16 in all our experiments. The dropout rate in the input layer was set to 0.5.

6.8 Results on Larger Datasets

In this section, we provide results on three recently proposed datasets which are relatively larger than standard benchmark datasets (Cora/Citeseer/Pubmed). Specifically, 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 [4]. 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 in presented in Table 5 and the results are presented in Table 4. We observe that GraphMix(GCN) improves the results over GCN for all the three datasets. We note that we did minimal hyperparameter search for

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

GraphMix(GCN) as mentioned in Section 6.8.1, and doing more rigorous hyperparameter search can further improve the performance of GraphMix .

Table 4: 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 5: Dataset statistics

Datasets	**Classes**	**Features**	**Nodes**	**Edges**
Cora-Full	67	8710	18703	62421
Coauthor-CS	15	6805	18333	81894
Coauthor-Physics	5	8415	34493	247962

6.8.1 Hyperparameter Details for Results in Table 4

For all the experiments we use the standard architecture mentioned in Section 6.7 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 consistency coefficient γ (line 13 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).