Latent Adversarial Training of Graph Convolution Networks

Hongwei Jin [1] **Xinhua Zhang** [1]

Abstract

Despite the recent success of graph convolution networks (GCNs) in modeling graph structured data, its vulnerability to adversarial attacks have been revealed and attacks on both node feature and graph structure have been designed. Direct extension of adversarial sample based defense algorithms meets with immediate challenge because computing the adversarial network requires substantial cost. We propose addressing this issue by perturbing the latent representations in GCNs, which not only dispenses with adversarial network generation, but also attains improved robustness and accuracy by respecting the latent manifold of the data. Experimental results confirm the superior performance over strong baselines.

1. Introduction

Neural networks have achieved great success on Euclidean data for image recognition, machine translation, and speech recognition, etc. However, modeling with non-Euclidean data—such as complex networks with geometric information and structural manifold—is more challenging in terms of data representation. Mapping non-Euclidean data to Euclidean, which is also referred to as embedding, is one of the most prevalent techniques. Recently, graph convolutional networks (GCNs) have received increased popularity in machine learning for structured data. The first phenomenal work of GCN was presented by (Bruna et al., 2013), which developed a set of graph convolutional operations based on spectral graph theory. The conventional GCN was introduced by (Kipf & Welling, 2017) for the task of node classification, and they represent nodes by repeated multiplication of augmented normalized adjacency matrix and feature matrix, which can be interpreted as the first order approximation of localized spectral filters on graphs (Hammond et al., 2011; Defferrard et al., 2016). GCNs have been widely applied to a variety of machine learning tasks, including node classification (Kipf & Welling, 2017), graph clustering (Duvenaud et al., 2015), link prediction (Kipf & Welling, 2016; Schlichtkrull et al., 2018), recommender systems (Berg et al., 2018), etc.

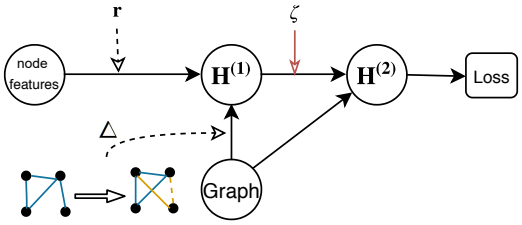

Figure 1. Different perturbation models for GCNs

Due to the recursive neighborhood expansion across layers, computation and memory are often required in considerable amount for GCNs, especially when the graph is large and dense. To alleviate these demands, Chen et al. (2018) proposed a batched scheme by introducing the Monte Carlo sampling across neighborhood. Similarly, Chen et al. (2017) introduced a stochastic version of GCNs and reduced the variance in the mean time. Wu et al. (2019) demonstrated that linearized GCNs do not impact the accuracy in many applications, while the resulting model can be easily scaled to large graphs naturally.

Despite the advantages in efficient and effective learning of representations and predictions, GCNs have been shown vulnerable to adversarial attacks. Although adversarial learning has achieved significant progress in recent years (Szegedy et al., 2014), the graph structure in GCNs contributes an additional layer of vulnerability. The conventional approaches based on adversarial samples (a.k.a. attacks) are typically motivated by adding imperceptible perturbation to images, followed by enforcing the invariance of prediction outputs (Kurakin et al., 2017). This corresponds to perturbing the node features in GCNs as shown by $\mathbf{r}$ in Figure 1, and has received very recent study. Feng et al. (2019) introduced the graph adversarial training (GAT) as a dynamic regularization scheme based on the graph structure. Deng et al. (2019) proposed a sample-based batch virtual adversarial training to promote the smoothness of the model.

However, the graph topology itself can be subject to attacks such as adding or deleting edges or nodes, as marked by

[1]Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA. Correspondence to: Hongwei Jin <hjin25@uic.edu>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data

Δ in Figure 1. Zügner et al. (2018) and Dai et al. (2018) constructed effective structural attacks both at training time (poisoning) and at testing time (evasion). Finding the adversarial input graph is indeed a combinatorial optimization problem that is typically NP-hard, and the perturbations are no longer imperceptible. Dai et al. (2018) proposed a reinforcement learning based attack that learns a generalizable attack policy to mis-classify a target in both graph classification and node classification. Zügner et al. (2018) introduced a surrogate model to approximate the perturbed graph structure and feature. Both of their methods considered attacks at the test stage, i.e., evasion attacks. By solving a bilevel problem from meta learning, Zügner & Günnemann (2019) generated adversarial examples that are graph structures, and in contrast to the previous attacks, they do not need to specify the target. Their attack modifies the training data to worsen the performance, hence a poison attack.

Although the technique of adversarial sample can be directly applied to defend against structural attacks, an immediate obstacle arises from computational complexity. All the aforementioned structural attacks are much more computation intensive than standard attacks in image classification. Therefore alternating between model optimization and adversarial sample generation can be prohibitively time consuming, making a new solution principle in high demand.

The goal of this paper, therefore, is to develop a new adversarial training algorithm that defends against attacks on both node features *and* graph structure, while at the same time maintaining or even improving the generalization accuracy. Our intuition draws upon two prior works. Firstly, in adversarial training over word inputs, Miyato et al. (2017) noted that words are discrete and are not amenable to infinitesimal perturbation. So they resorted to perturbing the word embeddings that are continuous.

A straightforward analogy of word embeddings in GCNs is the first layer output $\mathbf{H}^{(1)}$ after graph convolution, which blends the information of both node features and the graph. So we propose injecting adversarial perturbations to $\mathbf{H}^{(1)}$, as shown by ζ in Figure 1. This leads to indirect perturbations to the graph, which implicitly enforces robustness to structural attacks. As shown in Section 2.1, this can be achieved via a regularization term on $\mathbf{H}^{(1)}$, completely circumventing the requirement of generating adversarial attacks on the graph structure. We will refer to the approach as latent adversarial training (LAT-GCN).

However, Miyato et al. (2017) also noted that "the perturbed embedding does not map to any word" and "we thus propose this approach exclusively as a means of regularizing a text classifier". To address the analogous concern in GCNs, we leverage the observation by Stutz et al. (2019) that adversarial examples can benefit both robustness and accuracy if they are on the manifold of low-dimension embeddings. Using $\mathbf{H}^{(1)}$ as a proxy of the latent manifold, LAT-GCN manages to generate "on-manifold" perturbations, which, as our experiments show in Section 3, help to reduce the success rate of adversarial attacks for GCNs while preserving or lifting the accuracy of the model.

Notation. We denote the set of vertices for a graph G as $\mathcal{V}$, and denote the feature matrix as $\mathbf{X} \in \mathbb{R}^{n\times d}$, where $n = |\mathcal{V}|$ and d is the number of node features. $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix, and it is a $\{0, 1\}$ valued matrix for unweighted graphs. After adding a self looped link to each node, we have $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and we construct a diagonal matrix with $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$. The augmented normalized adjacency matrix is then defined as $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}$.

2. Latent Adversarial Training of GCNs

The original GCN model has the forward propagation as:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \quad l \geq 0, \tag{1}$$

where the initial node representation is $\mathbf{H}^0 = \mathbf{X}$. For notational convenience, we assume all nodes are represented by a d-dimensional vector in all layers, i.e., $\mathbf{W}^{(l)} \in \mathbb{R}^{n\times d}$. Without loss of generality, let us consider a two-layer GCN, which is commonly used in practice. Then the standard GCN tries to find the optimal weights $\boldsymbol{\theta} := (\mathbf{W}^{(1)}, \mathbf{W}^{(2)})$ that minimize some loss $f_{\boldsymbol{\theta}}$ (e.g., cross-entropy loss) over the latent representation $\mathbf{H}^{(2)}$. That is, $\min_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(G, \mathbf{X})$.

In order to improve the robustness to the perturbation in input feature $\mathbf{X}$, Feng et al. (2019) and Deng et al. (2019) considered *generative adversarial training* over $\mathbf{X}$, which at a high level, optimizes $\min_{\boldsymbol{\theta}} \max_{\mathbf{r}\in C} f_{\boldsymbol{\theta}}(G, \mathbf{X} + \mathbf{r})$. Here $\mathbf{r}$ is the perturbation chosen from a constrained domain C.

Naturally, it is also desirable to defend against attacks on the graph topology of G. Such structural attacks have been studied in (Dai et al., 2018; Zügner et al., 2018; Zügner & Günnemann, 2019), but no corresponding defense algorithm has been proposed yet. Different from attacks on $\mathbf{X}$, here the attacks on G are discrete (hence not imperceptible), and finding the most effective attacks can be NP-hard. This creates new obstacles to generative adversarial training, and therefore we resort to a regularization approach based on the *latent layer* $\mathbf{H}^{(1)}$.

Specifically, considering that the information in G and $\mathbf{X}$ is summarized into the first layer output $\mathbf{H}^{(1)}$, we can adopt generative adversarial training directly on $\mathbf{H}^{(1)}$:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\zeta}\in D} f_{\boldsymbol{\theta}}(\mathbf{H}^{(1)} + \boldsymbol{\zeta}), \tag{2}$$

where the symbol $f_{\boldsymbol{\theta}}$ is overloaded to denote the loss based on $\mathbf{H}^{(1)}$ with perturbation $\boldsymbol{\zeta}$. The benefits are two folds. Firstly, $\mathbf{H}^{(1)}$ is continuously valued, making it meaningful to apply small perturbations to it, which indirectly represent the perturbations in $\mathbf{X}$ *and* G. Secondly, Stutz et al. (2019) contended that perturbation at latent layers (such as $\mathbf{H}^{(1)}$)

Algorithm 1 Latent Adversarial Optimization for GCN

input $\mathbf{A}, \mathbf{X}$
  while not converged for (3) **do**
    while not converged for (5) **do**
      Apply ADAM to find $\boldsymbol{\zeta}^*$ (gradient in $\boldsymbol{\zeta}$ from Eq (6)).
    Take one step of ADAM in $\boldsymbol{\theta}$ with the gradient computed by $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{H}^{(1)}) + \gamma \nabla_{\boldsymbol{\theta}} \left\| \tilde{\mathbf{A}} \boldsymbol{\zeta}^* \mathbf{W}^{(2)} \right\|_F^2$.

are more likely to generate "on-manifold" samples that additionally benefits generalization, while perturbations in the raw input space (e.g., $\mathbf{X}$) is likely to deviate from the original data manifold and hence irrelevant to generalization.

Unfortunately, the perturbation $\boldsymbol{\zeta}$ in (2) is chosen *jointly* over *all* nodes in the graph setting, which is different from the common adversarial setting where each individual example seeks its own perturbation independently. As a result, the computational cost is higher, which is further exacerbated by the nested min-max optimization. To alleviate this problem, we further adopt the standard regularization variant of adversarial training, which aims to promote the smoothness of model predictions with respect to the perturbations:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\tilde{\mathbf{A}}, \mathbf{X}) := f_{\boldsymbol{\theta}}(\mathbf{H}^{(1)}) + \gamma \mathcal{R}_{\boldsymbol{\theta}}(\mathbf{H}^{(1)}), \quad (3)$$

where $\gamma \geq 0$ is a trade-off parameter, and the regularizer $\mathcal{R}_{\boldsymbol{\theta}}$ is defined as the Frobenius distance between the original model output (second layer) and that after perturbing $\mathbf{H}^{(1)}$:

$$\mathcal{R}_{\boldsymbol{\theta}}(\mathbf{H}^{(1)}) = \max_{\boldsymbol{\zeta} \in D} \left\| \tilde{\mathbf{A}} \left(\mathbf{H}^{(1)} + \boldsymbol{\zeta} \right) \mathbf{W}^{(2)} - \tilde{\mathbf{A}} \mathbf{H}^{(1)} \mathbf{W}^{(2)} \right\|_F^2. \quad (4)$$

Here we constrain the perturbed noise to be imperceptible via $D := \{\boldsymbol{\zeta} : \|\boldsymbol{\zeta}_{i:}\| \leq \epsilon, \forall i \in \{1, \ldots, n\}\}$. Although it is still a min-max problem, the inner maximization problem in $\mathcal{R}_{\boldsymbol{\theta}}$ is now decoupled with the loss $f_{\boldsymbol{\theta}}$, hence solvable with much ease. Indeed, both the objective and the constraint are quadratic, permitting efficient computation of gradient and projection. We also note in passing that turning the adversarial objective (2) into a regularized objective (3) is a commonly adopted technique, and their relationship has been studied by, e.g., Shafieezadeh-Abadeh et al. (2017).

2.1. Optimization

Since the objective (3) intrinsically couples all nodes, we apply ADAM to find the optimal $\boldsymbol{\theta}$ (Kingma & Ba, 2015), using the entire dataset as the mini-batch. The major complexity stems from $\nabla_{\boldsymbol{\theta}} \mathcal{R}_{\boldsymbol{\theta}}(\mathbf{H}^{(1)})$, which can be readily computed using the Danskin's theorem (Bertsekas, 1995). To this end, we first simplify (4) into

$$\mathcal{R}_{\boldsymbol{\theta}}(\mathbf{H}^{(1)}) = \max_{\boldsymbol{\zeta}} \left\| \tilde{\mathbf{A}} \boldsymbol{\zeta} \mathbf{W}^{(2)} \right\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{\zeta}_{i:}\| \leq \epsilon. \quad (5)$$

	citeseer	cora	pubmed
GCN	70.9 ± .5	81.4 ± .4	79.0 ± .4
FastGCN	68.8 ± .6	79.8 ± .3	77.4 ± .3
SGCN	70.8 ± .1	81.7 ± .5	79.0 ± .4
SGC	71.9 ± .1	81.0 ± .0	78.9 ± .0
LAT-GCN	72.1 ± .4	82.3 ± .3	78.8 ± .7

Table 1. Test accuracy for different models

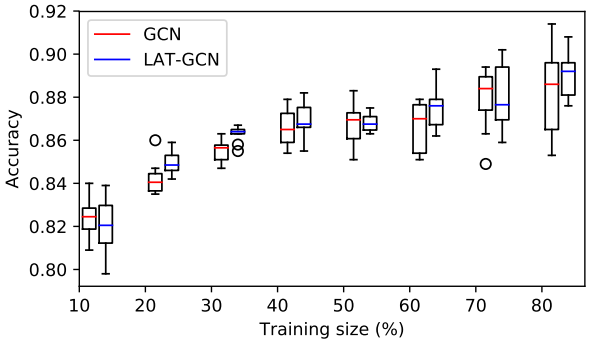

Figure 2. Test accuracy of GCN and LAT-GCN on Cora under varied size of training data.

Once we find the optimal $\boldsymbol{\zeta}^*$, the gradient in $\boldsymbol{\theta}$ is simply $\nabla_{\boldsymbol{\theta}} \left\| \tilde{\mathbf{A}} \boldsymbol{\zeta}^* \mathbf{W}^{(2)} \right\|_F^2$. To find $\boldsymbol{\zeta}$, note that the gradient in $\boldsymbol{\zeta}$ is

$$\nabla_{\boldsymbol{\zeta}} \operatorname{tr}(\tilde{\mathbf{A}} \boldsymbol{\zeta} \mathbf{W} \boldsymbol{\zeta}^\top \tilde{\mathbf{A}}^\top) = \left(\mathbf{W}^\top \boldsymbol{\zeta} + \mathbf{W} \boldsymbol{\zeta}^\top \right) \tilde{\mathbf{A}} \tilde{\mathbf{A}}^\top, \quad (6)$$

where $\mathbf{W} = \mathbf{W}^{(2)} \mathbf{W}^{(2)\top}$. Although the constraint $\|\boldsymbol{\zeta}_{i:}\| \leq \epsilon$ is convex and projection to it is trivial, the objective *maximizes* a convex function. So we simply use ADAM to approximately solve for $\boldsymbol{\zeta}$. The empirical results show that the additional optimization won't cost too much energy. For example, the walltime of each epoch on Cora dataset, which has less than 3000 nodes, would increase from 0.3 to 0.7 seconds on average, with 8-core CPU only. The overall procedure is summarized in Algorithm 1.

3. Experiments

We tested the performance of LAT-GCN on a range of standard citation datasets for node classification, including Citeseer, Cora, pubmed (Sen et al., 2008), cora_ml, dblp, and polblogs. The competing baselines include the vanilla GCN, FastGCN (Chen et al., 2018), SGCN (Chen et al., 2017), and SGC (Wu et al., 2019). All hyperparameters in respective models follow from the original implementation, including step size, width of layers, etc. Since the optimal objective value in (5) is quadratic in ϵ, only the value of $\gamma \epsilon^2$ matters for LAT-GCN. So we fixed $\gamma = 0.1$, and only tuned ϵ.

Finally, all algorithms are applied in a tranductive setting, where the graph is constructed by combining the nodes for training and testing. Accordingly, the perturbation $\boldsymbol{\zeta}$ is applied to both training and test nodes in (4).

Comparison of accuracy. We first compared the test accuracy as shown in Table 1. Each test is based on randomly

Figure 3. Test accuracy of LAT-GCN on Cora as a function of ϵ

Algorithm 2 Evaluation of Robustness

input $\mathbf{A}, \mathbf{X}, \Delta$ (budget of Nettack)
$s_{\text{GCN}} = s_{\text{LAT-GCN}} = 0$
$\mathcal{T}$:= sample 100 nodes from test set to attack
for $u \in \mathcal{T}$ (target of attack) & $g \in \{\text{GCN}, \text{LAT-GCN}\}$ **do**
  $c_g := evaluate(u, g(\mathbf{A}, \mathbf{X}))$
  if $c_g = c_{true}$ **then**
    $\mathbf{A}', \mathbf{X}' \leftarrow Nettack(\mathbf{A}, \mathbf{X}, u, \Delta)$
    $s_g := s_g + \delta(c_g \neq evaluate(u, g(\mathbf{A}', \mathbf{X}')))$
output success rates: $s_{\text{GCN}}/\left|\mathcal{T}\right|, s_{\text{LAT-GCN}}/\left|\mathcal{T}\right|$.

partitioning the nodes into training and testing for 20 times. Clearly, LAT-GCN delivers higher or similar accuracy compared with all the competing algorithms. Figure 2 compares LAT-GCN with GCN under varied size of training data from Cora, with another 10% for validation and the remaining nodes used for testing. It can be observed that LAT-GCN has better or competitive performance with the original GCN.

In order to study the influence of ϵ on the performance of LAT-GCN, we plotted in Figure 3 how the test accuracy changes with the value of ϵ. We again used the Cora dataset with 10% for training. Interestingly, the accuracy tends to increase as ϵ grows from 0 to 0.17, and then drops as ϵ grows further. This is consistent with the observation in (Stutz et al., 2019) where robustness is shown to be positively correlated with generalization, as long as the data points are not perturbed away from the latent manifold. More detailed results on other training sizes and on the Citeseer dataset are available in Appendix A (Figures 7 and 8).

Comparison of robustness. We next evaluated the robustness of LAT-GCN under Nettack, an evasion attack proposed by Zügner et al. (2018). A description of Nettack is available in Appendix A. The metric is the success rate across four datasets, with 100 randomly sampled nodes from the test set used as the attack target. Since Nettack can either attack structure only, or attack both structure and feature $\mathbf{X}$, we will refer to our results as LAT-GCN-A and LAT-GCN-AX, respectively. ϵ is set to 0.1 for LAT-GCN.

We first compared GCN with LAT-GCN when the perturbation budget Δ for Nettack was increased from 1 to 10. The test procedure is detailed in Algorithm 2. As shown in Figure 4 for Cora, LAT-GCN enjoys significantly lower success rate than GCN under the two different attack strategies. We did not compare with other variants because none of them is designed for defending against structural attacks.

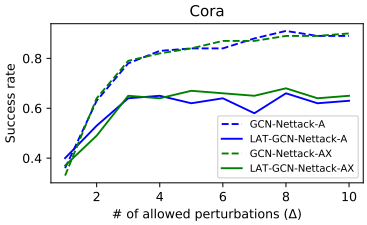

Figure 4. Success rate on Cora with increasing value of Δ

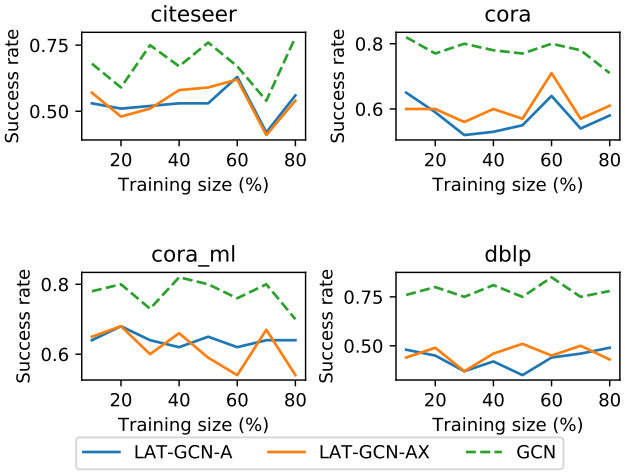

Figure 5. Success rate under varied training size in percentage

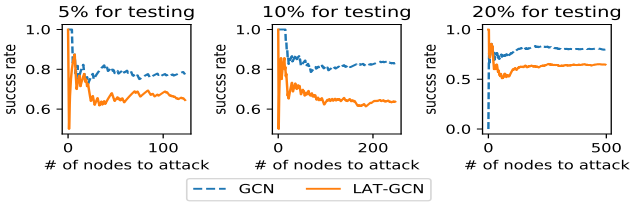

Figure 6. Success rate on Cora with varied test set size. 10% of the dataset was used for validation and the remaining for training.

The success rates on other datasets are provided in Figure 5, where the training set size is increased from 10% to 80%. Here we sampled 100 nodes from the test set to attack one after one. Finally, in order to study the influence of test set size, Figure 6 shows the success rate on Cora when the test set size is varied in 5%, 10%, and 20%, and *all* test nodes were targeted for attack. In both figures, the budget Δ was set to the degree of each targeted node. Clearly, LAT-GCN is significantly more robust than GCN.

Conclusion In this work, we proposed a new regularization technique for GCN which not only improves generalization, but also defends against attacks in both node feature and graph structure. The method, which is based on perturbing latent representations, can be extended to adversarial learning in dynamic graphs and multi-modality graphs.

References

Berg, R. v. d., Kipf, T. N., and Welling, M. Graph convolutional matrix completion. In *Knowledge Discovery and Data Mining*, 2018.

Bertsekas, D. P. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs, 2013.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. 2017.

Chen, J., Ma, T., and Xiao, C. Fastgcn: fast learning with graph convolutional networks via importance sampling. 2018.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. In *International Conference on Machine Learning*, pp. 1123–1132, 2018.

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

Deng, Z., Dong, Y., and Zhu, J. Batch virtual adversarial training for graph convolutional networks. *arXiv preprint arXiv:1902.09192*, 2019.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Feng, F., He, X., Tang, J., and Chua, T.-S. Graph adversarial training: Dynamically regularizing based on graph structure. *arXiv preprint arXiv:1902.08226*, 2019.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. *international conference on learning representations*, 2015.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

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

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. 2017.

Langley, P. Crafting papers on machine learning. In *Proc. Intl. Conf. Machine Learning*, pp. 1207–1212. Morgan Kaufmann, San Francisco, CA, 2000.

Miyato, T., Dai, A. M., and Goodfellow, I. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations (ICLR)*, 2017.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pp. 593–607. Springer, 2018.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Shafieezadeh-Abadeh, S., Kuhn, D., and Esfahani, P. M. Regularization via mass transportation. *arXiv preprint arXiv:1710.10016*, 2017.

Stutz, D., Hein, M., and Schiele, B. Disentangling adversarial robustness and generalization. *CVPR*, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Wu, F., Zhang, T., Souza Jr, A. H. d., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2847–2856. ACM, 2018.

Appendix A

Nettack

Nettack is the adversarial attack on graphs specifically designed for graph convolutional networks. The goal is to perform small perturbations on graph $G = (\mathbf{A}, \mathbf{X})$, leading to the new graph $G' = (\mathbf{A}', \mathbf{X}')$, such that the classification performance drops. The nettack limits the number of allowed changes by a budget:

$$\sum_u \sum_i \left| \mathbf{X}^{(0)}_{ui} - \mathbf{X}'_{ui} \right| + \sum_{u<v} \left| \mathbf{A}^{(0)}_{uv} - \mathbf{A}'_{uv} \right| \leq \Delta$$

In order to solve a discrete problem, they introduced the surrogate model, which is a linearized version of GCN,

$$\mathcal{L}_s(\mathbf{A}, \mathbf{X}; \mathbf{W}, v_o) = \max_{c \neq c_o} \left[\hat{\mathbf{A}}^2 \mathbf{X} \mathbf{W} \right]_{v_o, c} - \left[\hat{\mathbf{A}}^2 \mathbf{X} \mathbf{W} \right]_{v_o, c_o},$$

and aim to solve $\operatorname{argmax}_{(\mathbf{A}', \mathbf{X}')} \mathcal{L}_s(\mathbf{A}', \mathbf{X}'; \mathbf{W}, v_o)$, where v_o is the target node, and c_o is the original label of v_o.

However, this problem is still intractable to solve due to the discrete domain. To simplify, they define scoring functions that evaluate the surrogate loss obtained after adding/deleting a feature or edge. More specifically, they are:

$$\mathcal{S}_{struct}(e; G, v_o) := \mathcal{L}_s(\mathbf{A}', \mathbf{X}; \mathbf{W}, v_o),$$
$$\mathcal{S}_{feature}(f; G, v_o) := \mathcal{L}_s(\mathbf{A}, \mathbf{X}'; \mathbf{W}, v_o).$$

Hyperparameters

We examine how the hyperparameter ϵ affects the accuracy on the test set, under varied training set size in $\{10\%, 20\%, 30\%, 40\%\}$. The results for Cora and Citeseer are in Figures 7 and 8, respectively.

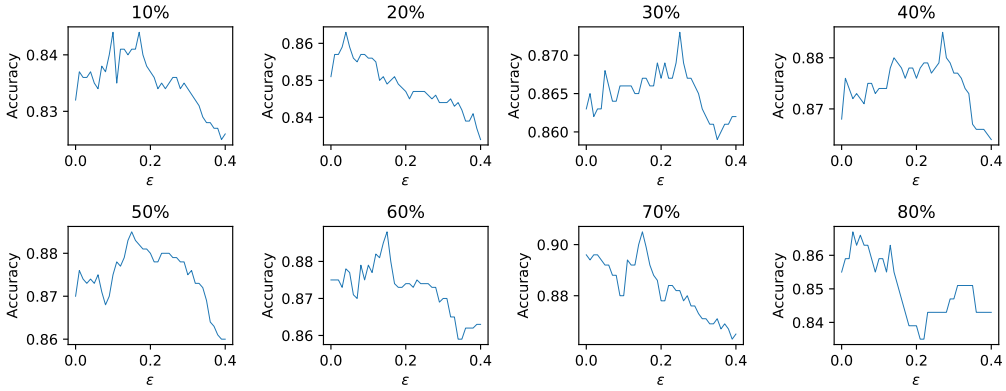

Figure 7. Test accuracy of LAT-GCN on Cora as a function of ϵ, over different sizes of training data.

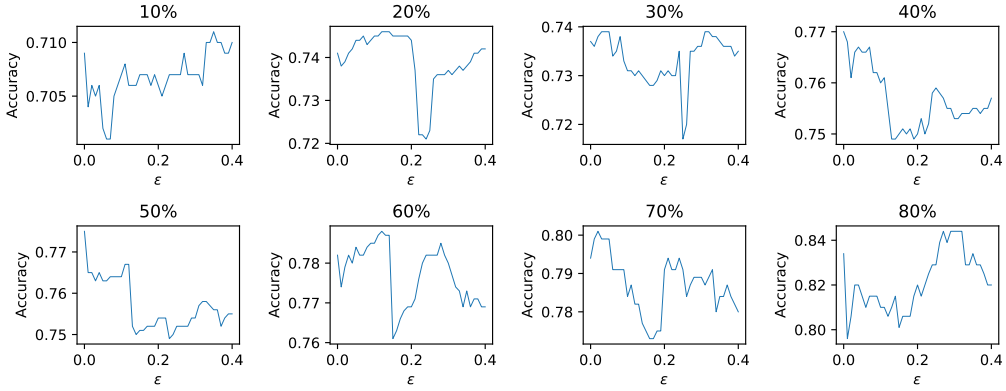

Figure 8. Test accuracy of LAT-GCN on Citeseer as a function of ϵ, over different sizes of training data.