Batch Virtual Adversarial Training for Graph Convolutional Networks

Zhijie Deng, Yinpeng Dong and Jun Zhu
Dept. of Comp. Sci. & Tech., State Key Lab of Intell. Tech. & Sys., TNList Lab,
Center for Bio-Inspired Computing Research, Tsinghua University, Beijing, 100084, China

Abstract

We present batch virtual adversarial training (BVAT), a novel regularization method for graph convolutional networks (GCNs). BVAT addresses the shortcoming of GCNs that do not consider the smoothness of the model's output distribution against local perturbations around the input. We propose two algorithms, sample-based BVAT and optimization-based BVAT, which are suitable to promote the smoothness of the model for graph-structured data by either finding virtual adversarial perturbations for a subset of nodes far from each other or generating virtual adversarial perturbations for all nodes with an optimization process. Extensive experiments on three citation network datasets *Cora*, *Citeseer* and *Pubmed* and a knowledge graph dataset *Nell* validate the effectiveness of the proposed method, which establishes state-of-the-art results in the semi-supervised node classification tasks.

1 Introduction

Recent models for graph-structured data (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018) demonstrate remarkable performance in semi-supervised node classification tasks. These methods essentially adopt different aggregators to aggregate feature information from the neighborhood of a node. The aggregators take the connectivity patterns and node features into consideration and enable the information propagation through edges (e.g., the gradient information can be distributed from nodes with labels to nodes without labels). By aggregating the features of a node with its nearby neighbors, the models can promote the smoothness between nodes in a neighborhood, which is helpful for semi-supervised node classification based on the assumption that connected nodes in the graph are likely to have similar representations (Kipf and Welling 2017). However, these methods only consider the smoothness between nodes in a neighborhood without considering the smoothness of the output distribution. Several studies have confirmed that smoothing the output distribution of neural networks (i.e., encourage the neural networks to produce similar outputs) against local perturbations around the input can improve the generalization performance in supervised and especially semi-supervised learning (Wager, Wang, and Liang 2013; Sajjadi, Javanmardi, and Tasdizen 2016; Laine and Aila 2017; Miyato et al. 2017; Luo et al. 2018). Moreover, it's crucial to encourage the smoothness of the output distribution for aggregator-based graph models since that the receptive field (e.g., Fig. 1a) of a single node grows exponentially with respect to the number of aggregators in a model (Chen and Zhu 2017), and neural networks tend to be non-smooth with such high dimensional input space (Goodfellow, Shlens, and Szegedy 2015; Peck et al. 2017). Therefore, it is necessary for aggregator-based graph models to encourage the **smoothness of their output distribution** in addition to the smoothness in the neighborhood.

In this work, we focus on graph convolutional networks (GCNs) (Kipf and Welling 2017), a typical and effective instance of aggregator-based graph learning models, and the proposed algorithms are generally applicable for other models such as graph attention networks (Veličković et al. 2018), GraphSAGE (Hamilton, Ying, and Leskovec 2017), etc. GCN model has fixed aggregators based on the adjacency matrix of input graph and is a special form of Laplacian smoothing (Li, Han, and Wu 2018). Considering an undirected graph[1] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, GCNs are built upon the symmetric sparse adjacency matrix $A \in \mathbb{R}^{N\times N}$. The input feature matrix of all nodes in $\mathcal{V}$ is denoted as $X \in \mathbb{R}^{N\times D}$ where D is the feature dimension of each node. GCNs use a normalized version of A as the propagation matrix P

$$\tilde{A} = A + I, \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \text{ and } P = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}.$$

In a K-layer GCN, each layer aggregates and processes information from itself and its nearby neighbors with the following propagation rule

$$H^{(0)} = X, \quad H^{(l+1)} = \sigma(PH^{(l)}W^{(l)}), \tag{1}$$

where $H^{(l)}$ is the activation matrix of the l-th layer, $W^{(l)}$ is the trainable transformation matrix and σ denotes an activation function. Particularly, the prediction of a node u depends on the input features of nodes in its receptive field (as shown in Fig. 1a), whose dimension grows exponentially with respect to the number of layers. The GCN model tends to be non-smooth with high dimensional input space, so it's necessary to add regularization terms to promote the smoothness

[1] GCNs represent a directed graph as an undirected bipartite graph with additional nodes that represent edges in the original graph.

of the model's output distribution against local perturbations around each input data point and improve the generalization performance (Sajjadi, Javanmardi, and Tasdizen 2016).

Virtual adversarial training (VAT) (Miyato et al. 2017) is one of the regularization methods that encourage the smoothness of the model in the input space. Instead of training the model to be isotropically smooth around each input, VAT defines the local distributional smoothness (LDS) at each input data point as the robustness of the model against the worst-case virtual adversarial perturbation. The virtual adversarial perturbation is approximated by the first dominant eigenvector of the Hessian matrix of the LDS loss. VAT has demonstrated competitive performance in semi-supervised image recognition (Miyato et al. 2017) and text classification tasks (Miyato, Dai, and Goodfellow 2017). However, the straightforward extension of VAT to semi-supervised node classification based on GCNs is less effective. We notice that GCNs are trained by a batch gradient descent algorithm, where all nodes in the graph are processed together for forward and backward propagation. So we expect to deploy VAT to calculate the LDS loss for all nodes at the same time instead of only one node in each training iteration, for the purpose of efficient training. In GCNs, the output distribution for a node u depends on the input features of nodes in its receptive field R_u (shown in Fig. 1a), and consequently, the gradient of u's loss can be back propagated to all nodes in R_u. It means that the virtual adversarial perturbation calculated for node u will modify the features of every node in R_u. On the other hand, for each node $v \in R_u$, it may also lie in the receptive field of other nodes rather than u, so the overall perturbation for v is the accumulation of the virtual adversarial perturbations of all nodes, in whose receptive field node v lies. As a result, the overall perturbation for R_u is actually not the worst-case virtual adversarial perturbation for node u, making the LDS loss inefficient to encourage the smoothness of the model's output distribution and unable to push decision boundaries of the model away from real data instances effectively. There are also some works concentrating on applying mini-batch training into GCNs (Hamilton, Ying, and Leskovec 2017; Chen and Zhu 2017), but the mini-batch training has the same problem as the whole batch training because of the possible nodes' interaction within each batch.

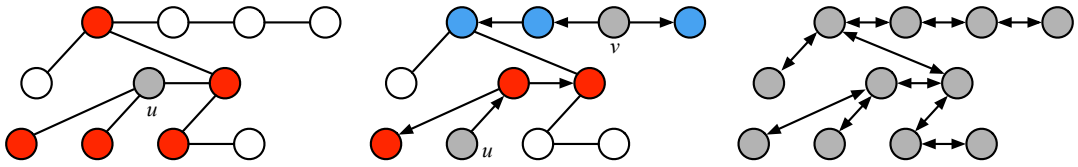
Given the aforementioned issue, we aim to generate virtual adversarial perturbations **perceiving the connectivity patterns between nodes in the graph** so that the LDS loss really reflects the worst-case smoothness at every nodes. Based on this motivation, we propose batch virtual adversarial training (BVAT) algorithms in this paper, which are appropriate for GCNs to promote the smoothness of the output distribution at all nodes. On one hand, we present a sample-based BVAT algorithm (S-BVAT) to craft local virtual adversarial perturbations for a subset of nodes, which are not interacted with each other. For K-layer GCNs, the receptive field of a node contains all the k-hop neighbors where $0 \leq k \leq K$. If two nodes u and v are selected, we expect that R_u doesn't have intersection with R_v, as shown in Fig. 1b. So the number of nodes in the shortest path between u and v should be at least $2K$ and we sample a subset of nodes based on this. We randomly sample a fixed number of nodes in each training iteration and calculate the LDS loss at the sampled nodes for training. On the other hand, we propose an optimization-based BVAT algorithm (O-BVAT) that generates adversarial perturbations by maximizing the LDS loss at all nodes together, as shown in Fig. 1c. Due to the interaction between nodes, it's difficult to generate such perturbations based on the second-order Taylor approximation of the LDS loss as used in VAT. Therefore, we solve this problem by taking an optimization approach, which proves to be more powerful in adversarial attacks (Carlini and Wagner 2017). Similarly we aggregate the LDS loss at all nodes for training.

BVAT has the same theoretical basis with VAT and they are both designed to satisfy the clustering assumption of classifiers. So theoretically speaking, GCNs with BVAT will have improved robustness, generalization performance and classification accuracy than GCNs. To empirically validate the effectiveness of BVAT, we conduct experiments on four challenging node classification benchmarks: *Cora*, *Citeseer*, *Pubmed* citation datasets as well as a knowledge graph dataset *Nell*. BVAT establishes state-of-the-art results across all datasets with a tolerable additional computation complexity. We also make extensive comparisons between BVAT and VAT for graph-structured data and show the superiority of BVAT.

2 Related Work

Learning node representations based on graph for semi-supervised learning and unsupervised learning has drawn increasing attention and has been developed mainly toward two directions: spectral approaches and non-spectral approaches. On one hand, label propagation (Zhu, Ghahramani, and Lafferty 2003), manifold regularization (Belkin, Niyogi, and Sindhwani 2006), deep semi-supervised embedding (Weston et al. 2012), Chebyshev expansion based spatially localized filters (Defferrard, Bresson, and Vandergheynst 2016) and graph convolutional networks (Kipf and Welling 2017) inherit the ideas from spectral graph theory (Ekambaram et al. 2013) and demonstrate impressive results in the context of node classification. On the other hand, non-spectral approaches learn graph embeddings directly on spatially local neighborhoods. DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and its variants (Tang et al. 2015; Grover and Leskovec 2016) learn node representations based on the neighborhood generated by random walks. Planetoid (Yang, Cohen, and Salakhudinov 2016), MoNet (Monti et al. 2017) and GraphSAGE (Hamilton, Ying, and Leskovec 2017) propose end-to-end frameworks for learning embeddings for semi-supervised learning or unsupervised learning on graph. Recently, graph attention networks (Veličković et al. 2018) introduce masked self-attentional layers into graph convolutions and establish a strong baseline for transductive and inductive learning on graph.

There is also an interest in applying regularization terms to semi-supervised learning based on the cluster assumption (Chapelle and Zien 2005). Various sophisticated solutions have been proposed (Miyato et al. 2017; Laine and Aila 2017; Tarvainen and Valpola 2017; Luo et al. 2018), which achieve striking results. Among them, virtual adversarial training (VAT) has been proved successful in the context of



(a) Receptive Field of node u (b) Perturbations for nodes in S-BVAT (c) Perturbations for nodes in O-BVAT

Figure 1: (a) The receptive field (marked by red) of a node u in two-layer GCNs. (b) In S-BVAT, two nodes u and v are selected to calculate the LDS loss, and the virtual adversarial perturbations are applied to the features in their receptive fields (marked by red and blue), which do not have intersection. (c) In O-BVAT, all nodes are included to calculate the LDS loss and the virtual adversarial perturbations for all nodes are optimized together.

semi-supervised image and text classification tasks (Miyato, Dai, and Goodfellow 2017; Miyato et al. 2017). However, VAT is not effective enough when straightforwardly applied to the models that deal with graph-structured data because of the interrelationship between different node instances. For this reason, we propose BVAT in the paper which introduces a novel regularization term to smooth the output distribution of the models.

3 Batch Virtual Adversarial Training

In this section, we first extend virtual adversarial training (VAT) (Miyato et al. 2017) to graph convolutional networks (GCNs) and discuss its shortcomings. We then propose the batch virtual adversarial training (BVAT) algorithms which are more suitable for GCNs. In particular, we introduce the sample-based batch virtual adversarial training (S-BVAT) and the optimization-based batch virtual adversarial training (O-BVAT) respectively.

3.1 Virtual Adversarial Training for Graph Convolutional Networks

Virtual adversarial training (VAT) (Miyato et al. 2017) encourages the smoothness by training the model to be robust against local worst-case virtual adversarial perturbation. In VAT, the local distributional smoothness (LDS) is defined by a virtual adversarial loss as

$$\mathrm{LDS}(x, \mathcal{W}, r_{\mathrm{vadv}}) = D_{\mathrm{KL}}\big(p(y|x, \hat{\mathcal{W}})||p(y|x+r_{\mathrm{vadv}}, \mathcal{W})\big), \tag{2}$$

where $p(y|x, \mathcal{W})$ is the posterior distribution parameterized by $\mathcal{W}$ (i.e., trainable parameters), $D_{\mathrm{KL}}(\cdot, \cdot)$ is the KL divergence of two distributions, $\hat{\mathcal{W}}$ denotes the current estimation of the parameters $\mathcal{W}$ and r_{vadv} is the virtual adversarial perturbation found by

$$\begin{aligned} r_{\mathrm{vadv}} &= \arg\max_{r;||r||_2\leq\epsilon} \mathrm{LDS}(x, \mathcal{W}, r) \\ &= D_{\mathrm{KL}}\big(p(y|x, \hat{\mathcal{W}})||p(y|x+r, \mathcal{W})\big). \end{aligned} \tag{3}$$

The closed form solution for r_{vadv} does not exist. In (Miyato et al. 2017), $\mathrm{LDS}(x, \mathcal{W}, r)$ is approximated by the second-order Taylor expansion and r_{vadv} emerges as the first dominant eigenvector of the Hessian matrix of $\mathrm{LDS}(x, \mathcal{W}, r)$ with respect to r.

When applying VAT to GCNs for semi-supervised node classification tasks, a straightforward extension of the LDS loss at each node is defined as $\mathrm{LDS}(X_u, \mathcal{W}, r_{\mathrm{vadv},u})$, where X_u is the input feature matrix of all nodes in u's receptive field R_u. In training GCNs, we use the average LDS loss for all nodes as a regularization term

$$\mathcal{R}_{\mathrm{vadv}}(\mathcal{V}, \mathcal{W}) = \frac{1}{N}\sum_{u\in\mathcal{V}} \mathrm{LDS}(X_u, \mathcal{W}, r_{\mathrm{vadv},u}). \tag{4}$$

So the overall training objective function is

$$\mathcal{L} = \mathcal{L}_0 + \alpha \cdot \frac{1}{|\mathcal{V}|}\sum_{u\in\mathcal{V}} E\big(p(y|X_u, \mathcal{W})\big) + \beta \cdot \mathcal{R}_{\mathrm{vadv}}(\mathcal{V}, \mathcal{W}), \tag{5}$$

where $E(\cdot)$ is the conditional entropy of a distribution. The conditional entropy of the output distribution for all nodes in $\mathcal{V}$ serves as an additional regularization term to encourage one-hot predictions, which is widely used in semi-supervised classification tasks (Grandvalet and Bengio 2005). α and β are coefficients for conditional entropy and local distributional smoothness. $\mathcal{L}_0$ is the average cross-entropy loss of nodes in the labeled nodes set $\mathcal{V}_{\mathcal{L}}$:

$$\mathcal{L}_0 = \frac{1}{|\mathcal{V}_{\mathcal{L}}|}\sum_{u\in\mathcal{V}_{\mathcal{L}}} \ell\big(y_u, p(y|X_u, \mathcal{W})\big). \tag{6}$$

However, we notice that a major difference between VAT on graph-structured data and other forms of data (e.g., images, texts) is that the prediction of one node relies on the input features of others, indicating that the virtual adversarial perturbation generated for one node will modify the features of other nodes in its receptive field. And if we generate virtual adversarial perturbations for all nodes in one iteration for efficient training, the perturbation applied to a node is actually the accumulation of the virtual adversarial perturbations generated by many nodes, in whose receptive field this node lies. So the resultant perturbation is **not** the worst-case virtual adversarial perturbation (which we will validate in Sec. 4.2). Therefore, the virtual adversarial loss $\mathrm{LDS}(X_u, \mathcal{W}, r_{\mathrm{vadv},u})$ may not reflect the robustness of the model's output distribution and cannot effectively encourage the local smoothness at each input instance.

3.2 Sample-based Batch Virtual Adversarial Training

In this section and the following section, we elaborate the proposed batch virtual adversarial training (BVAT) algorithms including the sample-based BVAT (S-BVAT) and the

optimization-based BVAT (O-BVAT). BVAT can perceive the connectivity patterns between nodes and alleviate the interaction effect of virtual adversarial perturbations crafted for all nodes by either stochastically sampling a subset of nodes far from each other or adopting a more powerful optimization process for generating virtual adversarial perturbations. These two approaches are both harmonious with the batch gradient descent optimization method used by GCNs and only increase several times of forward propagation and back propagation at each training iteration. The additional computation complexity is tolerable as shown in Sec. 4.5.

The motivation of sample-based BVAT is that we expect to make the model be aware of the relationship between nodes and limit the propagation of adversarial perturbations to prevent perturbations from different nodes interacting with each other. In S-BVAT, we generate virtual adversarial perturbations for a subset $\mathcal{V}_\mathcal{S} \subset \mathcal{V}$ of nodes, whose receptive fields do not overlap with each other. Taking a K-layer GCN model for example, the receptive field R_u of a node u contains all the k-hop neighbors where $0 \leq k \leq K$. If we expect R_u doesn't have intersection with R_v, the number of nodes in the shortest path between u and v (denoted as the distance D_{uv}) should be at least $2K$ (shown in Fig. 1b). Therefore, we randomly sample a subset $\mathcal{V}_\mathcal{S}$ of nodes with a fixed size B as

$$\mathcal{V}_\mathcal{S} = \{u | u \in \mathcal{V}\}, \text{ s.t. } |\mathcal{V}_\mathcal{S}| = B, \ \forall u, v \in \mathcal{V}_\mathcal{S}, D_{uv} \geq 2K.$$

In this way, the generated virtual adversarial perturbations for nodes in $\mathcal{V}_\mathcal{S}$ do not interact with each other. The regularization term for training is the average LDS loss over nodes in $\mathcal{V}_\mathcal{S}$ as

$$\mathcal{R}_{\text{vadv}}(\mathcal{V}_\mathcal{S}, \mathcal{W}) = \frac{1}{B} \sum_{u \in \mathcal{V}_\mathcal{S}} D_{\text{KL}}\big(p(y|X_u, \hat{\mathcal{W}}) \\ ||p(y|X_u + r_{\text{vadv},u}, \mathcal{W})\big), \quad (7)$$

where $\mathcal{R}_{\text{vadv}}(\mathcal{V}_\mathcal{S}, \mathcal{W})$ can be seen as an estimation of $\mathcal{R}_{\text{vadv}}(\mathcal{V}, \mathcal{W})$. $r_{\text{vadv},u}$ is the matrix of virtual adversarial perturbation in the same size as X_u, and is applied to the nodes in u's receptive field. $r_{\text{vadv},u}$ can be found by

$$\begin{aligned} r_{\text{vadv},u} &= \underset{r;||r||_F \leq \epsilon}{\arg\max} \, \text{LDS}(X_u, \mathcal{W}, r) \\ &= D_{\text{KL}}\big(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + r, \mathcal{W})\big). \end{aligned} \quad (8)$$

Similar to VAT (Miyato et al. 2017), we approximate $\text{LDS}(X_u, \mathcal{W}, r)$ with the second-order Taylor expansion and find the first dominant eigenvector of the Hessian matrix of $\text{LDS}(X_u, \mathcal{W}, r)$ with respect to r as $r_{\text{vadv},u}$ by a power iteration method (Golub and Van der Vorst 2001) with T iterations. Note that the virtual adversarial perturbations for all nodes in $\mathcal{V}_\mathcal{S}$ can be processed at the same time since that GCNs are trained by a batch gradient descent algorithm. And that's why we call the proposed algorithms batch virtual adversarial training. We use R to denote the perturbations of all nodes in the graph, which has the same size as X.

The overall objective function for training semi-supervised node classification GCNs is

$$\mathcal{L} = \mathcal{L}_0 + \alpha \cdot \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} E\big(p(y|X_u, \mathcal{W})\big) + \beta \cdot \mathcal{R}_{\text{vadv}}(\mathcal{V}_\mathcal{S}, \mathcal{W}), \quad (9)$$

Algorithm 1 Sample-based batch virtual adversarial training (S-BVAT)

1: $\mathcal{V}_\mathcal{S} = \varnothing$, $\mathcal{V}_\mathcal{C} = \mathcal{V}$.
2: **while** $|\mathcal{V}_\mathcal{S}| < B$ **do**
3: Choose a node u from $\mathcal{V}_\mathcal{C}$ randomly and add u to $\mathcal{V}_\mathcal{S}$.
4: Remove all nodes in the k-hop ($\forall k \in [0, 2K]$) neighborhood of u from $\mathcal{V}_\mathcal{C}$.
5: Initialize $r_{\text{vadv},u}$ from an iid Gaussian distribution and normalize it as $\|r_{\text{vadv},u}\|_F = 1$.
6: **end while**
7: Calculate $r_{\text{vadv},u}$ by taking the gradient of $\text{LDS}(X_u, \mathcal{W}, r)$ with respect to r:

$$g_u \leftarrow \nabla_r D_{\text{KL}}\big(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + r, \mathcal{W})\big)|_{r=\xi r_{\text{vadv},u}},$$

$$r_{\text{vadv},u} = \epsilon \cdot g_u / \|g_u\|_F.$$

8: **return** $\nabla_\mathcal{W} \mathcal{R}_{\text{vadv}}(\mathcal{V}_\mathcal{S}, \mathcal{W})|_{\mathcal{W}=\hat{\mathcal{W}}}$.

where $\mathcal{L}_0$ is the supervised loss defined in Eq. (6) and $E(\cdot)$ is the conditional entropy of a distribution. α and β are hyperparameters to balance the loss terms. We summarize S-BVAT in Algorithm 1. As suggested by (Miyato et al. 2017) and our experiments, one-step power iteration is sufficient for approximating $r_{\text{vadv},u}$ and obtaining high performance. So we simply use $T = 1$ in algorithm 1 and we further discuss the effect of T in Sec. 4.2.

3.3 Optimization-based Batch Virtual Adversarial Training

In the perspective of batch training, Eq. (3) of VAT for GCN can be formalized as

$$\underset{R;||R_v||_2 \leq \epsilon, v \in \mathcal{V}}{\arg\max} \frac{1}{N} \sum_{u \in \mathcal{V}} D_{\text{KL}}\big(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + R_u, \mathcal{W})\big),$$

where R is the adversarial perturbations matrix for the whole feature matrix X and R_u denotes some rows of R that related to the prediction of node u. The VAT algorithm in Sec. 3.1 search for R using the second-order Taylor approximation of the average LDS loss but it has two problems: 1) the limitation $||R_v||_2 \leq \epsilon$ cannot be ensured; 2) R_v for each node v are not in the worst-case adversarial direction. These problems are caused by the interaction between the LDS loss at nearby nodes as discussed in Sec. 3.1.

To overcome the problems of VAT for GCN, in the optimization-based batch virtual adversarial training (O-BVAT) algorithm, we propose to generate virtual adversarial perturbations for all nodes in $\mathcal{V}$ by an optimization process, which proves to be more powerful in adversarial attacks than one-step gradient-based methods (Carlini and Wagner 2017). We maximize the average LDS loss with respect to the perturbation matrix R so that the neighborhood perturbations R_u of every node u are adversarial enough. At the same time, we punish the norm of R so that the perturbations are small enough. Specifically, we want to generate a perturbation R in the same size as X, which is applied to all nodes in $\mathcal{V}$ and

Algorithm 2 Optimization-based batch virtual adversarial training (O-BVAT)

1: Initialize $R^{(0)} \in \mathbb{R}^{N \times D}$ from an iid Gaussian distribution.
2: **for** i = 1 to T **do**
3: Calculate the gradient of Eq. (10) with respect to $R^{(i-1)}$ as $g^{(i)}$.
4: Use an Adam optimizer to perform gradient ascent as $R^{(i)} \leftarrow \text{Adam}(R^{(i-1)}, g^{(i)})$.
5: **end for**
6: $R \leftarrow R^{(T)}$.
7: **return** $\nabla_{\mathcal{W}} \mathcal{R}_{\text{vadv}}(\mathcal{V}, \mathcal{W})|_{\mathcal{W}=\hat{\mathcal{W}}}$.

is optimized by solving

$$\arg\max_{R} \frac{1}{N} \sum_{u \in \mathcal{V}} D_{\text{KL}}\big(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + R_u, \mathcal{W})\big) - \gamma \cdot \|R\|_F^2, \quad (10)$$

where $\|R\|_F$ is the Frobenius norm of R which makes the optimal perturbation have a small norm, and γ is a hyper-parameter to balance the loss terms in Eq. (10). We optimize R with an Adam (Kingma and Ba 2014) optimizer for T iterations. The overall objective is similar to Eq. (9), except that the regularization term is the average LDS loss over the entire node set $\mathcal{V}$ as $\mathcal{R}_{\text{vadv}}(\mathcal{V}, \mathcal{W})$. We summarize the O-BVAT method in Algorithm 2.

4 Experiments

We empirically evaluate the BVAT algorithms through experiments on different datasets. We first specify the experimental settings in Sec. 4.1. We then demonstrate the effectiveness of the BVAT algorithms by showing that the regularization term can well represent the worst-case local smoothness and the models trained by our methods are quite smooth against perturbations in Sec. 4.2. Based on the smoothness of the model's output distribution, the BVAT algorithms boost the performance of GCNs significantly and achieve superior results on four graph transductive learning benchmarks against a wide variety of state-of-the-art methods, which is detailed in Sec. 4.3. Moreover, we test the effects of hyper-parameters on the performance of BVAT in Sec. 4.4 and we talk about the additional computation complexity of BVAT in Sec. 4.5.

4.1 Experimental Setup

Datasets. We examine BVAT[2] on the three citation network datasets *Cora*, *Citeseer* and *Pubmed* (Sen et al. 2008) and one knowledge graph dataset *Nell* (Yang, Cohen, and Salakhudinov 2016). We closely follow the experimental setup of (Yang, Cohen, and Salakhudinov 2016) and (Kipf and Welling 2017) with the same train/validation/test splits. The details of the four datasets are summarized in Table 1. We use the same preprocessing strategies as GCNs. The dimension of the preprocessed node features in *Nell* is $61,278$,

[2]We slightly abuse BVAT for both the algorithms and the GCN models trained with BVAT when no ambiguity is possible.

	Cora	Cireseer	Pubmed	Nell
Nodes	2,708	3,327	19,717	65,755
Edges	5,429	4,732	44,338	266,144
Features	1,433	3,703	500	61,278
Classes	7	6	3	105
Label rate	0.052	0.036	0.003	0.001

Table 1: Statistics of the datasets used in our experiments.

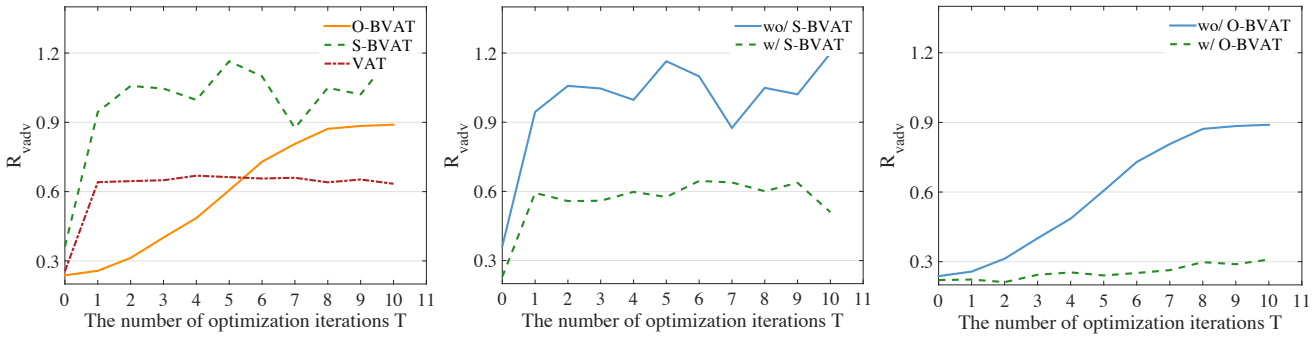
so the input sparse matrix $X \in \mathbb{R}^{65755 \times 61278}$ is too large to be converted to a dense matrix that doesn't exceed the GPU memory (GTX 1080Ti). As a result, BVAT algorithms construct sparse virtual adversarial perturbations R for *Nell*.

Configurations. We utilize the same architecture and initialization as the original 2-layer GCNs and keep the hyper-parameters of GCNs (dropout rate for two layers, L_2 regularization factor for the first layer, number of hidden units and number of epochs) unchanged. We only optimize the additional hyper-parameters of BVAT based on the generalization performance on validation sets of different benchmarks. For S-BVAT, we fix $B = 100$ according to the number of nodes and the sparsity of the four datasets. We fix $\xi = 10^{-6}$ and $T = 1$. We set the perturbation size $\epsilon = 0.03$ for *Cora* and *Citeseer* and $\epsilon = 0.003$ for *Pubmed* and *Nell*. We tune the hyper-parameters α and β because the label rate, feature size and number of edges vary significantly across different datasets. For O-BVAT, we run an Adam optimizer with learning rate 0.001 for $T = 10$ iterations. We set $\gamma = 0.01$ on *Pubmed* and $\gamma = 1$ on the other three datasets. We also tune the hyper-parameters α, β for different datasets.

4.2 Effectiveness of BVAT

We evaluate the effectiveness of the BVAT algorithms by assessing the virtual adversarial perturbations generated by them for graph data. First, we train a baseline two-layer GCN model on *Cora* without any other regularization except dropout and weight decay. Second, we use VAT, S-BVAT and O-BVAT to manufacture virtual adversarial perturbations through power iteration method (VAT and S-BVAT) or optimization-based method (O-BVAT) respectively. In VAT, the virtual adversarial perturbations are generated for all the nodes as introduced in Sec. 3.1. Finally, we use the resultant perturbations to calculate the regularization term $\mathcal{R}_{\text{vadv}}$ averaged by the local distribution smoothness among all the nodes $\mathcal{V}$ (in VAT and O-BVAT) or a subset of nodes $\mathcal{V}_{\mathcal{S}}$ (in S-BVAT). $\mathcal{R}_{\text{vadv}}$ indicates whether the perturbations are worst-case locally adversarial or not. We plot how $\mathcal{R}_{\text{vadv}}$ changes with respect to the number of iterations T of these three algorithms in Fig. 2a.

It is clear that O-BVAT and S-BVAT achieve higher $\mathcal{R}_{\text{vadv}}$ values than VAT, which demonstrates that BVAT can find virtual adversarial perturbations which are more likely to be in the worst-case direction and more effective and helpful for virtual adversarial training. For both S-BVAT and VAT, We can observe a significant jump of the $\mathcal{R}_{\text{vadv}}$ value when T changes from 0 (random perturbations) to 1, but cannot observe much more improvement of the $\mathcal{R}_{\text{vadv}}$ with larger T.



(a) (b) (c)

Figure 2: (a) Comparisons of $\mathcal{R}_{\text{vadv}}$ of VAT, S-BVAT and O-BVAT on a baseline GCN model. (b) Comparisons of $\mathcal{R}_{\text{vadv}}$ of S-BVAT on models trained with and without S-BVAT. (c) Comparisons of $\mathcal{R}_{\text{vadv}}$ of O-BVAT on models trained with and without O-BVAT.

Method	**Cora**	**Cireseer**	**Pubmed**	**Nell**
ManiReg (Belkin, Niyogi, and Sindhwani 2006)	59.5	60.1	70.7	21.8
SemiEmb (Weston et al. 2012)	59.0	59.6	71.1	26.7
LP (Zhu, Ghahramani, and Lafferty 2003)	68.0	45.3	63.0	26.5
DeepWalk (Perozzi, Al-Rfou, and Skiena 2014)	67.2	43.2	65.3	58.1
Planetoid (Yang, Cohen, and Salakhudinov 2016)	75.7	64.7	77.2	61.9
Monet (Monti et al. 2017)	81.7 ± 0.5	–	78.8 ± 0.3	–
GAT (Veličković et al. 2018)	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	–
GCN (Kipf and Welling 2017)	81.5	70.3	79.0	66.0
GCN w/ random perturbations	82.3 ± 2.0	71.4 ± 1.9	79.2 ± 0.6	65.9 ± 1.0
GCN w/ VAT	82.8 ± 0.8	73.0 ± 0.7	79.5 ± 0.3	66.0 ± 1.1
GCN w/ S-BVAT	83.4 ± 0.6	73.1 ± 1.3	79.6 ± 0.5	66.0 ± 0.9
GCN w/ O-BVAT	**83.6** ± 0.5	**74.0** ± 0.6	**79.9** ± 0.4	**67.1** ± 0.6

Table 2: Summary of node classification results in terms of test accuracy (%).

Therefore we simply set $T = 1$ in the following experiments of VAT and S-BVAT. However, $\mathcal{R}_{\text{vadv}}$ of O-BVAT increases monotonically with respect to T and converges at a much bigger value than that of random perturbations after 10 steps, so we choose $T = 10$ in the following node classification experiments.

On the other hand, we compare the robustness (i.e., smoothness in the worst-case direction) of the models trained with S-BVAT/O-BVAT with that of the models trained without S-BVAT/O-BVAT by calculating the $\mathcal{R}_{\text{vadv}}$ values. Fig. 2b and 2c show the results respectively. As we have observed, the models trained with BVAT (S-BVAT or O-BVAT) have lower $\mathcal{R}_{\text{vadv}}$ values than the vanilla GCN models, which indicates that the models trained with BVAT are more robust against the adversarial perturbations and more smooth in the input space. The smoothness of the model's output distribution will in consequence improve the generalization performance, as showed in the next section.

4.3 Semi-supervised Node Classification

BVAT can decrease model's $\mathcal{R}_{\text{vadv}}$ effectively and improve model's smoothness and robustness against local perturbations. Therefore, BVAT should have improved generalization performance on node classification tasks. To empirically validate this, we deploy BVAT and VAT algorithms for semi-supervised node classification on the *Cora*, *Citeseer*, *Pubmed* and *Nell* datasets, and compare the accuracy of the trained models with several state-of-the-art methods in Table 2. We also train a GCN model with random input perturbations as a baseline.

To keep comparisons fair, we adopt the same network architecture, initialization and hyper-parameters as GCNs and report the averaged results of 10 runs with different random seeds. ± in Table 2 represents the standard deviation of test accuracy of 10 runs.

The proposed GCNs with VAT, GCNs with S-BVAT and GCNs with O-BVAT all outperform the vanilla GCNs and GCNs trained with random input perturbations by a large margin across all the four datasets. Furthermore, as expected, O-BVAT boosts the performance significantly and GCNs trained with O-BVAT establish state-of-the-art results in these semi-supervised node classification tasks. When comparing the performance between S-BVAT and O-BVAT, we find that O-BVAT demonstrates better performance because the O-BVAT algorithm uses the local distributional smoothness at all nodes as regularization for training, which may be

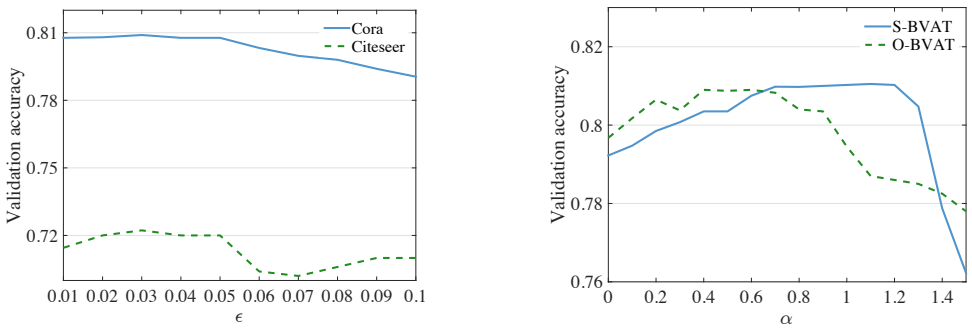


Figure 3: (a) Effect of ϵ on the validation performance of S-BVAT for *Cora* and *Citeseer*. (b) Effect of α on the validation performance of S-BVAT and O-BVAT for *Cora*.

more efficient than using local distributional smoothness at a subset of nodes. We also notice that the performance of GCNs trained with VAT is only a little worse than GCNs trained with S-BVAT. The reason is that though the perturbations produced by GCNs trained with VAT are not in the worst-case direction, they are adversarial and could help to improve smoothness to some extent. In one iteration, GCNs with VAT make use of all the nodes in the graph which are more than B nodes used by GCNs with S-BVAT to do virtual adversarial training, so it can explore the input space more sufficiently to meet the clustering assumption of classifier.

4.4 Ablation study of ϵ and α

To investigate how the perturbation size ϵ in S-BVAT affects the final classification results, we conduct an ablation study in Fig. 3a, where we plot the validation accuracy of the trained models on *Cora* and *Citeseer* with respect to the varied ϵ while keeping the other hyper-parameters fixed (on *Cora*, we set $\beta = 1.2$ and $\alpha = 0.7$; on *Citeseer*, we set $\beta = 0.8$ and $\alpha = 0.7$). As we have observed, S-BVAT is not sensitive to ϵ when it changes from 0.01 to 0.1 and we choose $\epsilon = 0.03$ for both *Cora* and *Citeseer*. The conclusion is also true for *Pubmed* and *Nell*, where ϵ is set to 0.003 due to the smaller norm of input features in these two datasets.

Incorporating the conditional entropy term $E(\cdot)$ into training of semi-supervised tasks is confirmed useful generally (Grandvalet and Bengio 2005). We expect to determine the importance of the role it plays in the BVAT algorithms. Thus, we conduct two set of experiments on *Cora* by assessing the validation accuracy of the models trained with different values of α in the range $[0, 1.5]$ with a granularity 0.1. The results of the two algorithms (we assign β to 1.2 and 1.5 for S-BVAT and O-BVAT respectively) are plotted in Fig. 3b. The results demonstrate that S-BVAT and O-BVAT can remain high performance with regularization coefficient α varying in a large range. Therefore, we think that the virtual adversarial perturbations used in BVAT play a crucial role in smoothing the output distribution of the model and improving its performance, while the conditional entropy term $E(\cdot)$ is an extra regularization which made the model more suitable for semi-supervised classification.

4.5 Computation Complexity Analysis

Actually BVAT algorithms will only bring a tolerable additional computation complexity because BVAT algorithms work in a batch manner and they only need to calculate the gradients of the LDS loss with respect to the input feature matrix without updating the parameters of the graph convolutional neural networks classifier. We empirically estimate the time consuming of GCN, GCN with S-BVAT and GCN with O-BVAT on *Cora* dataset, and they averagely need 0.0355154, 0.03946125 and 0.1064431 seconds for one epoch respectively. GCN with S-BVAT is a little slower than GCN as there are only two additional forward propagations and one additional back propagation. GCN with O-BVAT spends less than $3\times$ time than GCN because the optimization process involves T + 1 additional forward propagations and T additional back propagations (T = 10 in all the experiments). Considering that the classification performance of GCN W/ BVAT is noticeably better than GCN, its no doubt that the extra calculation cost is acceptable.

5 Conclusion

In this paper, we propose batch virtual adversarial training algorithms for learning graph convolutional networks, which can smooth the output distribution against local worst-case perturbations at each node in the graph and are essentially suitable for any aggregator-based graph neural networks. In particular, we present sample-based batch virtual adversarial training and optimization-based virtual adversarial training algorithms respectively, which address the issues of vanilla virtual adversarial training method applied to graph-structured data. Experimental results demonstrate the effectiveness of the proposed method on various datasets in the semi-supervised node classification task. BVAT outperforms the current state-of-the-art methods by a large margin.

References

[Belkin, Niyogi, and Sindhwani 2006] Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7(Nov):2399–2434.

[Carlini and Wagner 2017] Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 39–57. IEEE.

[Chapelle and Zien 2005] Chapelle, O., and Zien, A. 2005. Semi-supervised classification by low density separation. In *AISTATS*, 57–64. Citeseer.

[Chen and Zhu 2017] Chen, J., and Zhu, J. 2017. Stochastic training of graph convolutional networks. *arXiv preprint arXiv:1710.10568*.

[Defferrard, Bresson, and Vandergheynst 2016] Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NIPS)*, 3844–3852.

[Ekambaram et al. 2013] Ekambaram, V. N.; Fanti, G.; Ayazifar, B.; and Ramchandran, K. 2013. Wavelet-regularized graph semi-supervised learning. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, 423–426. IEEE.

[Golub and Van der Vorst 2001] Golub, G. H., and Van der Vorst, H. A. 2001. Eigenvalue computation in the 20th century. In *Numerical analysis: Historical developments in the 20th century*. Elsevier. 209–239.

[Goodfellow, Shlens, and Szegedy 2015] Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

[Grandvalet and Bengio 2005] Grandvalet, Y., and Bengio, Y. 2005. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 529–536.

[Grover and Leskovec 2016] Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864. ACM.

[Hamilton, Ying, and Leskovec 2017] Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 1025–1035.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

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

[Laine and Aila 2017] Laine, S., and Aila, T. 2017. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*.

[Li, Han, and Wu 2018] Li, Q.; Han, Z.; and Wu, X. M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*.

[Luo et al. 2018] Luo, Y.; Zhu, J.; Li, M.; Ren, Y.; and Zhang, B. 2018. Smooth neighbors on teacher graphs for semi-supervised learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Miyato et al. 2017] Miyato, T.; Maeda, S.-i.; Koyama, M.; and Ishii, S. 2017. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*.

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

[Monti et al. 2017] Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Peck et al. 2017] Peck, J.; Roels, J.; Goossens, B.; and Saeys, Y. 2017. Lower bounds on the robustness to adversarial perturbations. In *Advances in Neural Information Processing Systems (NIPS)*, 804–813.

[Perozzi, Al-Rfou, and Skiena 2014] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710. ACM.

[Sajjadi, Javanmardi, and Tasdizen 2016] Sajjadi, M.; Javanmardi, M.; and Tasdizen, T. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1163–1171.

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

[Tang et al. 2015] Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 1067–1077. International World Wide Web Conferences Steering Committee.

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

[Veličković et al. 2018] Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *International Conference on Learning Representations (ICLR)*.

[Wager, Wang, and Liang 2013] Wager, S.; Wang, S.; and Liang, P. S. 2013. Dropout training as adaptive regulariza-

tion. In *Advances in Neural Information Processing Systems (NIPS)*, 351–359.

[Weston et al. 2012] Weston, J.; Ratle, F.; Mobahi, H.; and Collobert, R. 2012. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer. 639–655.

[Yang, Cohen, and Salakhudinov 2016] Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning (ICML)*, 40–48.

[Zhu, Ghahramani, and Lafferty 2003] Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, 912–919.