

# Batch Virtual Adversarial Training for Graph Convolutional Networks

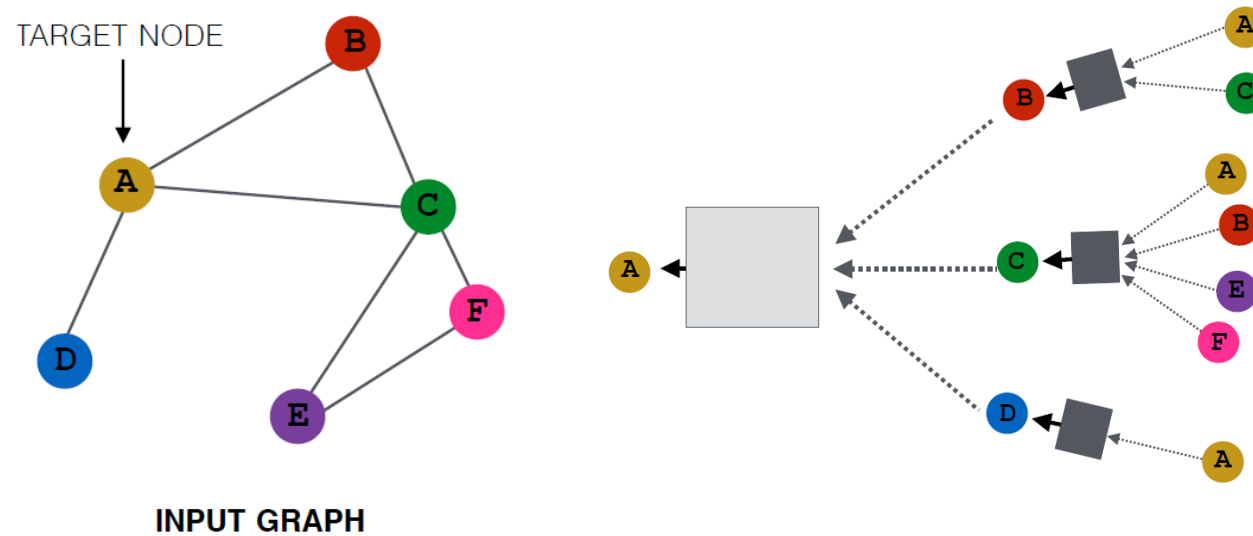
Zhijie Deng, Yinpeng Dong, Jun Zhu

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

Presented by Zhe Xu

# Graph Convolutional Networks (GCNs)

**Receptive Field (RF):** the set of nodes whose features are used as the input of GCN.



E.g. to predict the label of node A, if using a 1-layer GCN, the RF is {A, B, C, D}; if using a 2-layer GCN, the RF is {A, B, C, D, E, F}

# Challenges

- the receptive field (input space) of a single node grows exponentially with respect to the number of aggregators in the model
- neural network models tend to be non-smooth with high dimensional input space



- necessary to encourage the smoothness of the output distribution of existing graph models

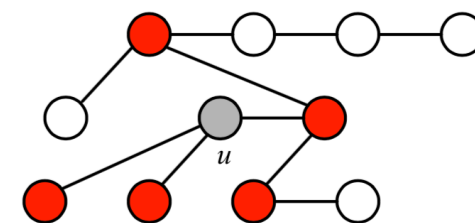
# A straightforward extension of VAT into GCNs in semi-supervised node classification task

**Given:** the graph topology, features of all the nodes, labels of part of nodes

**Output:** labels of the unlabeled nodes

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

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



(a) Receptive Field of node  $u$

**LDS : local distributional smoothness (loss)**

$$\text{LDS}(X_u, \mathcal{W}, r_{\text{vadv}}) = D_{KL}(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + r_{\text{vadv}}, \mathcal{W}))$$

$$\begin{aligned} r_{\text{vadv}} &= \arg \max_{||r||_F \leq \epsilon} \text{LDS}(X_u, \mathcal{W}, r) \\ &= \arg \max_{||r||_F \leq \epsilon} D_{KL}(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + r, \mathcal{W})) \end{aligned}$$

$\mathcal{V}$  denotes the node set of the graph containing  $N$  nodes.

$X_u$  is the input feature matrix of all nodes in  $RF$  of node  $u$ .

$r_{\text{vadv}, u}$  is a node  $u$ -specific perturbation matrix with same shape as matrix  $X_u$ .

$p(y|x, \mathcal{W})$  is the output distribution of the model given parameters  $\mathcal{W}$ .

$L_0$  is the loss function (e.g. cross-entropy) for labeled nodes.

$E(\cdot)$  is the conditional entropy of a distribution.

$\mathcal{W}$  is the set of parameters of the model.

$\hat{\mathcal{W}}$  is the current estimation of parameters  $\mathcal{W}$ .

$\hat{\mathcal{W}}$  and  $\mathcal{W}$  have no difference.

# Virtual Adversarial Training (VAT)

$$\text{LDS}(X_u, \mathcal{W}, r_{adv}) = D_{KL}(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + r_{adv}, \mathcal{W}))$$

$$\begin{aligned} r_{adv} &= \arg \max_{||r||_F \leq \epsilon} \text{LDS}(X_u, \mathcal{W}, r) \\ &= \arg \max_{||r||_F \leq \epsilon} D_{KL}(p(y|X_u, \hat{\mathcal{W}}) || p(y|X_u + r, \mathcal{W})) \end{aligned}$$

- A regularization method based on the priori knowledge that outputs of most naturally occurring systems are smooth with respect to spatial and/or temporal inputs. (the lower LDS is, the smoother the system is)
- Virtual adversarial direction: most greatly alter the output distribution in the sense of distributional divergence

# How to get $r$ under the setting of graph data?

Calculate  $r_{\text{vadv},u}$  by taking the gradient of  $\text{LDS}(X_u, \mathcal{W}, r)$  with respect to  $r$ :

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

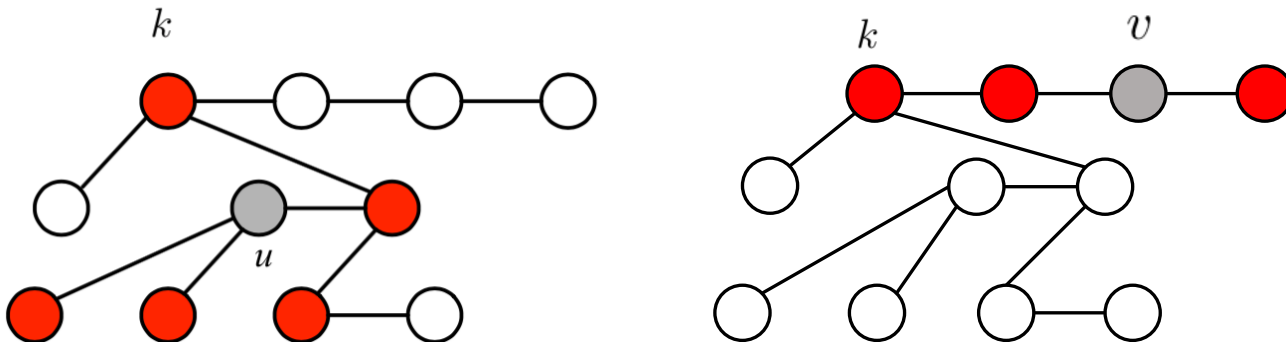
$$r_{\text{vadv},u} = \epsilon \cdot g_u / \|g_u\|_F.$$

**The perturbation is sample-specific which would be added on the feature matrix of nodes in RF of  $u$ .**

# Does the straightforward extension work?

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

In the batch training, the calculated  $r_{adv,u}$  is the accumulation of the virtual adversarial perturbations generated for all nodes in  $RF_u$ . Therefore,  $\|r_{adv,u}\| \leq \epsilon$  cannot be guaranteed and the resultant perturbations are not the worst-case virtual adversarial perturbations.



Example: for GCNs with **2** layers, if batch size = 2 and select node  $u$  and  $v$  in batch training, the perturbations on node **k** will accumulate twice, makes it far from the worst case.

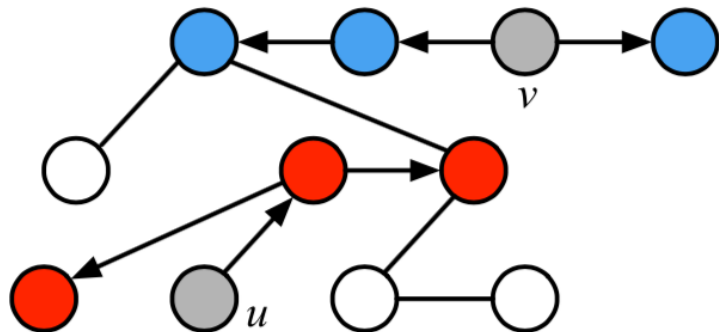
# Sample-based BVAT (S-BVAT)

In S-BVAT, they generate virtual adversarial perturbations for a subset  $\mathcal{V}_s \subset \mathcal{V}$  of nodes, whose receptive fields do not overlap with each other.

Taking a  $K$ -layer GCN model for example, the receptive field  $RF_u$  of a node  $u$  contains all the  $k$ -hop neighbors of it where  $0 \leq k \leq K$ . If we expect  $RF_u$  doesn't have intersection with  $RF_v$ , the number of nodes in the shortest path between  $u$  and  $v$  (denoted as the distance  $D_{uv}$ ) should be at least  $2K$ .

$$\mathcal{R}_{\text{adv}}(\mathcal{V}_s, \mathcal{W}) = \frac{1}{B} \sum_{u \in \mathcal{V}_s} \text{LDS}(X_u, \mathcal{W}, r_{\text{adv}, u}). \quad \mathcal{V}_s = \{u | u \in \mathcal{V}\}, \text{ s.t. } |\mathcal{V}_s| = B, \forall u, v \in \mathcal{V}_s, D_{uv} \geq 2K.$$

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



Example: 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.




# Optimization-based BVAT (O-BVAT)

Idea: Evaluate the regularization term for each samples in the training batch together and calculate a general perturbation matrix. (argmax is removed out of the summation)

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

$$R_{vadv}(\mathcal{V}, \mathcal{W}) = \frac{1}{N} \sum_{u \in \mathcal{V}} LDS(X_u, \mathcal{W}, R)$$

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

~~$$r_{vadv} := \arg \max_{r; ||r||_2 \leq \epsilon}$$~~

# Experiments

$R_{vadv}$  indicates whether the perturbations are worst-case locally adversarial or not. Figure 2.(a)

The models trained with BVAT (S-BVAT or O-BVAT) have lower  $R_{vadv}$  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. Figure 2.(b)/(c)

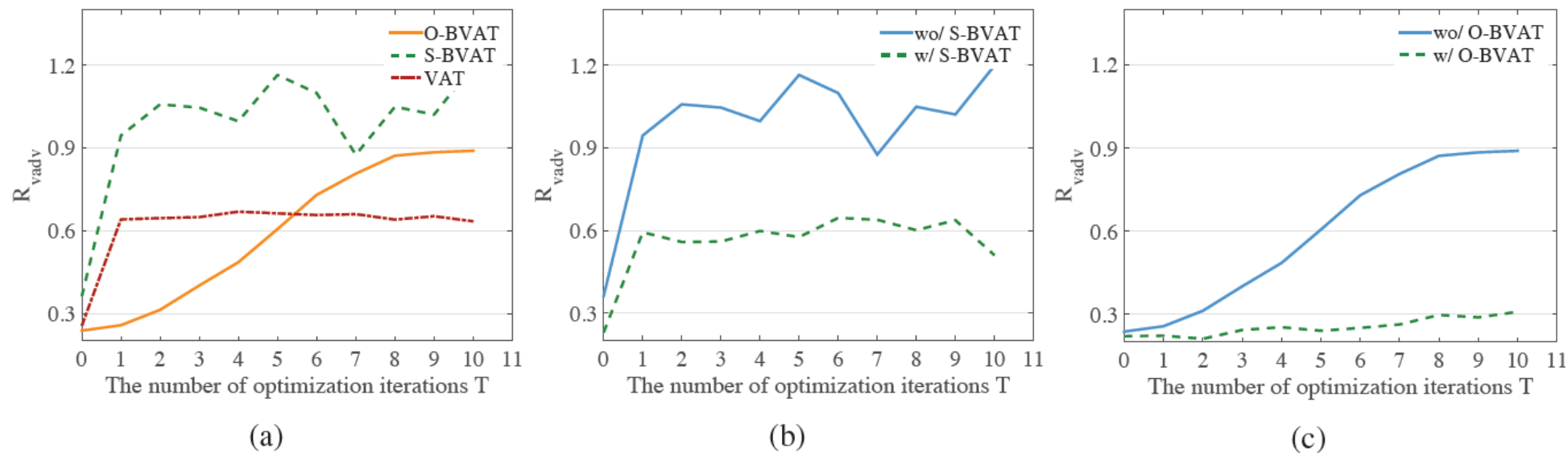


Figure 2. (a) Comparisons of  $\mathcal{R}_{vadv}$  of VAT, S-BVAT and O-BVAT on a baseline GCN model. (b) Comparisons of  $\mathcal{R}_{vadv}$  of S-BVAT on models trained with and without S-BVAT. (c) Comparisons of  $\mathcal{R}_{vadv}$  of O-BVAT on models trained with and without O-BVAT.

# Experiments

## Performance on the Semi-supervised Node Classification task

Table 1. Summary of node classification results in terms of test accuracy (%).

Method	Cora	Citeseer	Pubmed	Nell
ManiReg (Belkin et al., 2006)	59.5	60.1	70.7	21.8
SemiEmb (Weston et al., 2012)	59.0	59.6	71.1	26.7
LP (Zhu et al., 2003)	68.0	45.3	63.0	26.5
DeepWalk (Perozzi et al., 2014)	67.2	43.2	65.3	58.1
Planetoid (Yang et al., 2016)	75.7	64.7	77.2	61.9
Monet (Monti et al., 2017)	$81.7 \pm 0.5$	–	$78.8 \pm 0.3$	–
GAT (Veličković et al., 2018)	$83.0 \pm 0.7$	$72.5 \pm 0.7$	$79.0 \pm 0.3$	–
GPNN (Liao et al., 2018)	81.8	69.7	79.3	63.9
GCN (Kipf & Welling, 2017)	81.5	70.3	79.0	66.0
GCN w/ random perturbations	$82.3 \pm 2.0$	$71.4 \pm 1.9$	$79.2 \pm 0.6$	$65.9 \pm 1.0$
<b>GCN w/ VAT</b>	$82.8 \pm 0.8$	$73.0 \pm 0.7$	$79.5 \pm 0.3$	$66.0 \pm 1.1$
<b>GCN w/ S-BVAT</b>	$83.4 \pm 0.6$	$73.1 \pm 1.3$	$79.6 \pm 0.5$	$66.0 \pm 0.9$
<b>GCN w/ O-BVAT</b>	<b><math>83.6 \pm 0.5</math></b>	<b><math>74.0 \pm 0.6</math></b>	<b><math>79.9 \pm 0.4</math></b>	<b><math>67.1 \pm 0.6</math></b>

# Question: Why it is named 'Virtual' Adversarial Training?

## ---- A side by side comparison

**Intuition: The direction of the perturbation is the opposite direction of the 'right' direction.**

**In Adversarial Training the 'right' direction is the ground truth.**

**In Virtual Adversarial Training the 'right' direction is the current direction of the model.**

### Adversarial Training

$$L_{\text{adv}}(x_l, \theta) := D[q(y|x_l), p(y|x_l + r_{\text{adv}}, \theta)]$$

$$r_{\text{adv}} := \arg \max_{r; \|r\| \leq \epsilon} D[q(y|x_l), p(y|x_l + r, \theta)],$$

### Virtual Adversarial Training

$$\text{LDS}(x_*, \theta) := D[p(y|x_*, \hat{\theta}), p(y|x_* + r_{\text{vadv}}, \theta)]$$

$$r_{\text{vadv}} := \arg \max_{r; \|r\|_2 \leq \epsilon} D[p(y|x_*, \hat{\theta}), p(y|x_* + r)],$$

$q$  is the true distribution of labels (ground truth).

$p$  is the output distribution of model with parameter  $\theta$ .

$D$  is the objective function measures the divergence between two distributions.

$x_l$  are samples with labels.

$x_*$  are mixed samples (with or without labels).

# Deficiencies

- In the proposed O-BVAT, the norm of perturbations may not be constraint within a range.
- For S-BVAT, the max size of batch is limited. For a K-layer GCN, if the diameter of the graph is less than  $2K$ , then there doesn't exist suitable training batch.
- In the evaluation part of the proposed algorithms under the semi-supervised classification task, Graph Attention Networks may be a better baseline compared with GCNs.

# How to get r in classic scenarios? (optional)

**Now the input is feature vector but not feature matrix**

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

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

$$D(x, \mathcal{W}, r) = \text{LDS}(x, \mathcal{W}, r)$$

**Assume that  $D(\cdot)$  is twice differentiable w.r.t.  $\mathcal{W}$ ,  $x$ , and  $r$ .**

**It takes the minimal value 0 at  $r = 0$ , dictates that its first derivative w.r.t.  $r$  is zero when  $r = 0$ .**

$$D(x, \mathcal{W}, r) \approx \frac{1}{2} r^T H(x, \mathcal{W}) r \quad H(x, \mathcal{W}) := \nabla_r \nabla D(x, \mathcal{W}, r)|_{r=0}$$

$$\begin{aligned} r_{\text{vadv}} &\approx \arg \max_{r; ||r||_2 \leq \epsilon} \{r^T H(x, \mathcal{W}) r\} \\ &= \overline{\epsilon u(x, \mathcal{W})} \end{aligned}$$

where  $\bar{v}$  is the unit vector with same directions as vector  $v$ , and  $u(x, \mathcal{W})$  is the first dominant eigenvector of  $H(x, \mathcal{W})$ .

# References

Miyato, Takeru, et al. "Virtual adversarial training: a regularization method for supervised and semi-supervised learning." *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018): 1979-1993.

Deng, Zhijie, Yinpeng Dong, and Jun Zhu. "Batch Virtual Adversarial Training for Graph Convolutional Networks." *arXiv preprint arXiv:1902.09192* (2019).