2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 978-1-6654-6946-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CVPR52688.2022.00016

Robust Optimization as Data Augmentation for Large-scale Graphs

Kezhi Kong [1], Guohao Li [2], Mucong Ding [1], Zuxuan Wu [1], Chen Zhu [1],
Bernard Ghanem [2], Gavin Taylor [3], Tom Goldstein [1]

[1] University of Maryland, College Park
{kong, mcding, zxwu, chenzhu, tomg}@cs.umd.edu

[2] King Abdullah University of Science and Technology
{guohao.li, bernard.ghanem}@kaust.edu.sa

[3] US Naval Academy
taylor@usna.edu

Abstract

Data augmentation helps neural networks generalize better by enlarging the training set, but it remains an open question how to effectively augment graph data to enhance the performance of GNNs (Graph Neural Networks). While most existing graph regularizers focus on manipulating graph topological structures by adding/removing edges, we offer a method to augment node features for better performance. We propose FLAG (Free Large-scale Adversarial Augmentation on Graphs), which iteratively augments node features with gradient-based adversarial perturbations during training. By making the model invariant to small fluctuations in input data, our method helps models generalize to out-of-distribution samples and boosts model performance at test time. FLAG is a general-purpose approach for graph data, which universally works in node classification, link prediction, and graph classification tasks. FLAG is also highly flexible and scalable, and is deployable with arbitrary GNN backbones and large-scale datasets. We demonstrate the efficacy and stability of our method through extensive experiments and ablation studies. We also provide intuitive observations for a deeper understanding of our method. We open source our implementation at https://github.com/devnkong/FLAG.

1. Introduction

Graph Neural Networks (GNNs) have emerged as powerful architectures for learning and analyzing graph representations. The Graph Convolutional Network (GCN) [21] and its variants have been applied to a wide range of tasks, including visual recognition [33], meta-learning [11], social analysis [23, 29], and recommender systems [41]. However, the training of GNNs on large-scale datasets usually suffers from overfitting, and realistic graph datasets often involve a high volume of out-of-distribution test nodes [17], posing significant challenges for prediction problems.

One promising solution to combat overfitting in deep neural networks is data augmentation [22], which is commonplace in computer vision tasks. Data augmentations apply label-preserving transformations to the inputs, such as translations and reflections for images. As a result, data augmentation effectively enlarges the training set while incurring negligible computational overhead. However, it remains an open problem how to effectively generalize the notion of data augmentation to GNNs. Transformations on images rely heavily on image structures [3], and it is challenging to design low-cost transformations that preserve semantic meaning for non-visual tasks like natural language processing [38] and graph learning. Generally speaking, graph data for machine learning comes with graph structure (or edge features) and node features. In the limited cases where data augmentation can be done on graphs, it generally focuses exclusively on the graph structure by adding/removing edges [13, 14, 16, 30, 37, 42].

In the meantime, adversarial data augmentation, which applies small perturbations in the input feature space to maximially alter model outputs, is known to boost neural network robustness and promote resistance to adversarially chosen inputs [15, 26]. Despite the wide belief that adversarial training harms standard generalization and leads to worse accuracy [1, 35], recently a growing amount of attention has been paid to using adversarial perturbations to augment datasets and ultimately alleviate overfitting. For example, [36] and [34] showed adversarial data augmentation is a data-dependent regularization that could help generalize to out-of-distribution samples, and its efficacy has been verified in domains including computer vision [40], language understanding [19, 27, 44], and visual question answering [10]. Despite the success of adversarial augmenta-

978-1-6654-6946-3/22/$31.00 ©2022 IEEE
DOI 10.1109/CVPR52688.2022.00016

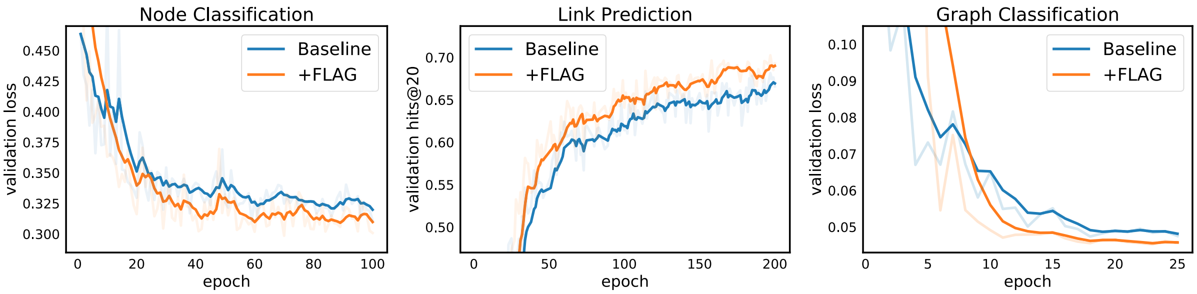


Figure 1. Generalization performance of FLAG on all three tasks. Left: node classification with GAT as baseline on `ogbn-products`; Middle: link prediction with hits@20 as metric (the higher the better) and GraphSAGE as baseline on `ogbl-ddi`; Right: graph classification with GIN as baseline on `ogbg-molhiv`. Plotted lines are attained by smoothing the original lines (the shallow ones), where smooth weights are 0.75, 0.75, and 0.5 respectively.

tion in language and vision, it remains unclear how to effectively and efficiently improve GNNs' clean accuracy using adversarial augmentation.

Present work. We propose **FLAG**, **F**ree **L**arge-scale **A**dversarial Augmentation on **G**raphs, to tackle the overfitting problem. While existing literature focuses on modifying graph structures to augment datasets, FLAG works purely in the node feature space by adding adversarial perturbations (generated by gradient-based robust optimization algorithms), to the input node features with graph structures unchanged. FLAG leverages "free" adversarial training methods [31] to conduct efficient adversarial training so that it is highly scalable to large datasets. The method also takes advantage of multi-scale adversarial augmentation to make the model fully generalized in the input feature space. We verify the effectiveness of our method on the *Open Graph Benchmark* (OGB) [17], which is a collection of large-scale, realistic, and diverse graph datasets for node, link, and graph property prediction tasks. We conduct extensive experiments across OGB datasets by applying FLAG to competitive GNN baselines and show that FLAG brings nontrivial improvements in most cases. For example, FLAG lifts the test accuracy of GAT on `ogbn-products` by an absolute value of 2.31%. FLAG is simple (easy to implement with a dozen lines of code in PyTorch), general (model-free and task-free), and efficient (able to bring salient improvement at tractable or even no extra cost). Our main contributions are summarized as follows:

- *Method:* To the best of our knowledge, our work is the *first* general-purpose feature-based data augmentation method on graph data, which is complementary to other regularizers (e.g., dropout) and topological augmentations. The novel method incorporates "free" and multi-scale techniques to craft feature augmentations more effectively.

- *Experiments:* We show the efficacy and scalability of our method through extensive experiments and ablation studies on large-scale datasets across node, link, and graph property prediction tasks. We validate that FLAG is superior to existing adversarial augmentation methods.

- *Analysis:* We provide observations and analysis to support our conjecture that the discrete vs. continuous distribution discrepancy of input features is the key to different effects (beneficial vs. harmful) of adversarial augmentations on model accuracy.

2. Preliminaries and Related Work

Graph Neural Networks (GNNs). We denote a graph as $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with initial node features x_v for $v \in \mathcal{V}$ and edge features e_{uv} for $(u, v) \in \mathcal{E}$. GNNs are built on graph structures to learn representation vectors h_v for every node $v \in \mathcal{V}$ and a vector $h_{\mathcal{G}}$ for the entire graph $\mathcal{G}$. Following [18], formally the k-th iteration of message passing, or the k-th layer of GNN forward path is defined as:

$$\begin{aligned} msg_v^{(k)} &= \text{AGGREGATE}_\theta^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}\right), \forall u \in \mathcal{N}(v)\right\}\right) \\ h_v^{(k)} &= \text{COMBINE}_\phi^{(k)}\left(h_v^{(k-1)}, msg_v^{(k)}\right), \end{aligned} \tag{1}$$

where $h_v^{(k)}$ is the embedding of node v at the k-th layer, e_{uv} is the feature vector of the edge between node u and v, $\mathcal{N}(v)$ is node v's neighbor set, and $h_v^{(0)} = x_v$. AGGREGATE$(\cdot)$ and COMBINE$(\cdot)$ functions are parameterized by neural networks.

To obtain the representation of the entire graph $h_{\mathcal{G}}$, the permutation-invariant READOUT$(\cdot)$ function pools node features from the final iteration K as:

$$h_{\mathcal{G}} = \text{READOUT}\left(\left\{h_v^{(K)} \mid v \in \mathcal{V}\right\}\right), \tag{2}$$

Existing graph regularizers mainly focus on augmenting graph structures by modifying edges [2, 16, 30]. GraphAT [8], BVAT [5], and LAT [20] are three *semi-supervised* methods on the node classification task. GraphAT promotes local smoothness by reinforcing the similarity between the predictions of perturbed nodes and their neighbors. BVAT proposed two graph VAT schemes to enhance the output smoothness of GCN; LAT virtually perturbed the first-layer embedding of a GCN classifier. The usage scenario of these methods is limited to node classification, while data augmentation should function regardless of tasks. Besides, the formulation of VAT [28] utilized by these works involves both supervised clean and adversarial robust losses simultaneously. Practically this will consume at least twice the GPU memory as the baseline, making them not scalable to large-scale datasets. Overall, no work so far has considered general-purpose feature-based data augmentations for large-scale graphs.

3. Proposed Method

In this work, we investigate how to effectively improve the generalization of GNNs through a feature-based augmentation. Graph node features are usually constructed as discrete embeddings, such as binary bag-of-words vectors or categorical variables. As a result, standard hand-crafted augmentations, like flipping and cropping transforms used in computer vision, are not applicable to graphs node features.

By hunting for and stamping out small perturbations that cause the classifier to fail, one may hope that adversarial training could benefit standard accuracy [15, 28, 35]. It is widely observed that when the data distribution is sparse and discrete, the beneficial effect of adversarial perturbations on generalization takes over [10, 35]. [36] viewed adversarial perturbation as a data-dependent regularization, which could intuitively generalize to out-of-distribution samples. Highlighted by [17], the out-of-distribution phenomenon of data is salient in the graph domain, and also considering the sparsity of labeled node samples in the semi-supervised node classification task, we view adversarial perturbation as a strong candidate method for input feature augmentation.

Min-Max Optimization. Adversarial training is the process of crafting adversarial data points, and then injecting them intro training data. This process is often formulated as the following min-max problem:

$$\min_{\boldsymbol{\theta}} \; E_{(x,y)\sim\mathcal{D}}\left[\max_{\|\boldsymbol{\delta}\|_p\leq\epsilon} L\left(f_{\boldsymbol{\theta}}(x+\boldsymbol{\delta}), y\right)\right], \tag{3}$$

where $\mathcal{D}$ is the data distribution, y is the label, $\|\cdot\|_p$ is some ℓ_p-norm distance metric, ϵ is the perturbation budget, and L is the objective function. [26] showed that this saddle-point optimization problem could be reliably tackled by Stochastic Gradient Descent (SGD) for the outer minimization and Projected Gradient Descent (PGD) for the inner maximization. In practice, the typical approximation of the inner maximization under an l_∞-norm constraint is as follows,

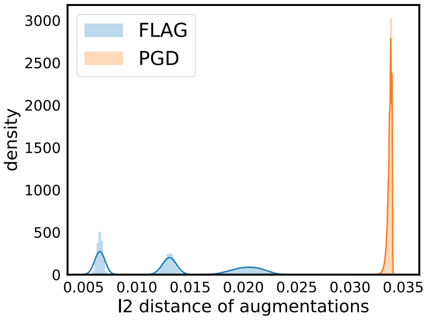


Figure 2. Augmentation distance distributions of FLAG and PGD. We run the test on `ogbn-arxiv` with GCN as backbone. Ascent steps are both set to 3.

$$\boldsymbol{\delta}_{t+1} = \Pi_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}\left(\boldsymbol{\delta}_t + \alpha\cdot\operatorname{sign}\left(\nabla_{\boldsymbol{\delta}} L\left(f_{\boldsymbol{\theta}}(x+\boldsymbol{\delta}_t), y\right)\right)\right), \tag{4}$$

where the perturbation $\boldsymbol{\delta}$ is updated iteratively, and $\Pi_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}$ performs projection onto the ϵ-ball in the l_∞-norm. For maximum robustness, this iterative updating procedure usually loops M times to craft the worst-case noise, which requires M forward and backward passes end-to-end. Afterwards the most vicious noise $\boldsymbol{\delta}_M$ is applied to the input feature, on which the model weight is optimized. The algorithm above is called PGD.

Multi-scale Augmentation. On visual tasks, [3] highlighted the importance of using *diverse* types of data augmentations such as *random cropping*, *color distortion*, and *Gaussian blur*. The authors showed that a single transformation is not sufficient to learn good representations. To fully exploit the generalizing ability and enhance the diversity and quality of adversarial perturbations, we propose to craft multi-scale augmentations. To realize this goal, we leverage the techniques below.

"Free" training. We leverage "free" adversarial training [31] to craft adversarial data augmentations. PGD is a powerful yet inefficient way of solving the min-max optimization. It runs M full forward and backward passes to craft a refined perturbation $\boldsymbol{\delta}_{1:M}$, but the model weights $\boldsymbol{\theta}$ only get updated once using the final $\boldsymbol{\delta}_M$. This process makes model training M times slower. In contrast, while computing the gradient for the perturbation $\boldsymbol{\delta}$, "free" training simultaneously produces the model parameter $\boldsymbol{\theta}$ on the same backward pass. This enables a parameter update to be computed in parallel with a perturbation update at virtually no additional cost. The authors proposed to train on the same minibatch M times in a row to simulate the inner maximization in Eq. (3), while compensating by performing M times fewer epochs of training. The resulting algorithm yields accuracy and robustness competitive with standard adversarial training, but with the same runtime as clean training.

Besides the efficiency, the "free" method achieves our idea of optimizing $\boldsymbol{\theta}$ with multi-scale augmentations. Note that X is augmented with additive perturbations $\boldsymbol{\delta}_{1:M}$, of which each can have a maximum scale of $m\alpha, m \in \{1, \cdots, M\}$, in contrast to PGD whose perturbation is a single $\boldsymbol{\delta}_M$ with an $M\alpha$ scaling. This greatly adds to the diversity of our augmentations. However, the "free" algorithm is suboptimal in terms of min-max optimization in that during the batch-replay process, the approximated perturbation computed to maximize the objective on $\boldsymbol{\theta_t}$ is used to robustly optimize $\boldsymbol{\theta_{t+1}}$ rather than $\boldsymbol{\theta_t}$. To tackle this problem, instead of directly updating $\boldsymbol{\theta}$ using the "by-product" gradient attained from the gradient ascent step on $\boldsymbol{\delta}$, we accumulate the gradients $\nabla_{\boldsymbol{\theta}} L$, and apply them to the model parameters all at once later. Formally, the optimization step is

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \frac{\tau}{M} \sum_{t=1}^{M} \nabla_{\boldsymbol{\theta}} L \left(f_{\boldsymbol{\theta}}(x + \boldsymbol{\delta_t}), y \right), \qquad (5)$$

where τ is learning rate and $\boldsymbol{\delta}_1$ is uniform noise. Note that the gradients in Eq.(5) are restored when crafting perturbation in Eq.(4). We save one backward pass and M times extra GPU memory through accumulating gradients (which is fully supported by PyTorch) during the batch replay process. Figure 2 depicts the effects of our design. We can see that PGD inevitably produces concentrated augmentations in terms of the magnitude, whereas our method produces perturbations with a broader range of sizes, which adds to the diversity and quality of the augmentations.

Moreover on the node classification task, we propose to augment labeled vs. unlabeled nodes with diverse magnitudes of perturbations during training time to further diversify the augmentations. We call it *Weighted perturbation*. When classifying one target node, messages from the whole k-hop neighborhood are aggregated and combined into its embedding. It is natural to believe that a further neighbor should have lower impact, i.e. higher smoothness, on the final decision of the target node, which can also be intuitively reflected by the recursive message passing procedure of GNNs in Eq.(1). In practice we find that a larger perturbation for unlabeled nodes can be beneficial to the performance. Algorithm 1 summarizes the pseudo code of our method on node classification task. Figure 1 illustrates the generalization ability of our proposed method.

4. Experiments

In this section, we conduct extensive experiments to fully reveal the efficacy of our method.

Datasets. We demonstrate FLAG's effectiveness through extensive experiments on the *Open Graph Benchmark* (OGB), which consists of a wide range of challenging large-scale datasets. [32], [7], and [6] showed that traditional graph datasets suffered from problems such as unrealistic and arbitrary data splits, highly limited data sizes, nonrigorous evaluation metrics, and common neglect of validation set, etc. In order to empirically study FLAG's effects in a fair and reliable manner, we conduct experiments on the OGB [17] datasets, which have tackled those major issues and brought more realistic challenges to the graph research community.

Algorithm 1 FLAG: **F**ree **L**arge-scale **A**dversarial Augmentation on **G**raphs (Node Classification Task)

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_l$ is the labeled node set; learning rate τ; ascent steps M; ascent step size α_v for labeled node, α_u for unlabeled, we assume the neighbors of labeled nodes are all unlabeled ones; $L(\cdot)$ as objective function; A$(\cdot)$ and C$(\cdot)$ denote the AGGREGATE and COMBINE functions in Eq.(1). The backward function at line 12 refers to back-propagation gradient computation for both model weights and noises.

```
 1: Initialize $(\theta, \phi)$
 2: for $v \in \mathcal{V}_l$ do
 3:     $\delta_v^{(0)} \leftarrow U(-\alpha_v, \alpha_v)$
 4:     $\delta_u^{(0)} \leftarrow U(-\alpha_u, \alpha_u)$
 5:     for t $= 1 \ldots M$ do
 6:         $h_v^{(0)} \leftarrow x_v + \delta_v^{(t-1)}$
 7:         $h_u^{(0)} \leftarrow x_u + \delta_u^{(t-1)}$
 8:         for $k = 1 \ldots K$ do
 9:             $msg_v^{(k)} \leftarrow \mathrm{A}_\theta^{(k)} \left( \left\{ \left( h_v^{(k-1)}, h_u^{(k-1)}, e_{uv} \right), \forall u \in \mathcal{N}(v) \right\} \right)$
10:             $h_v^{(k)} \leftarrow \mathrm{C}_\phi^{(k)} \left( h_v^{(k-1)}, msg_v^{(k)} \right)$
11:         end for
12:         $L \left( h_v^{(K)}, y \right)$ .backward()
13:         $g_{\theta,\phi}^{(t)} \leftarrow g_{\theta,\phi}^{(t-1)} + \frac{1}{M} \cdot \mathrm{grad}(\theta, \phi)$
14:         $\delta_v^{(t)} \leftarrow \delta_v^{(t-1)} + \alpha_v \cdot \mathrm{sign}\left(\mathrm{grad}\left(\delta_v\right)\right)$
15:         $\delta_u^{(t)} \leftarrow \delta_u^{(t-1)} + \alpha_u \cdot \mathrm{sign}\left(\mathrm{grad}\left(\delta_u\right)\right)$
16:     end for
17:     $(\theta, \phi) \leftarrow (\theta, \phi) - \tau \cdot g_{\theta,\phi}^{(M)}$
18: end for
```

Setup. FLAG drops the projection step when performing the inner maximization, in light of the positive effect of large perturbations on generalization [36], and also to simplify hyperparameter search. Usually on images, the inner maximization has a norm constraint on the perturbation; the largest perturbation one can add is bounded by the hyperparameter ϵ, typically 8/255 under the l_∞-norm. This ϵ encourages the visual imperceptibility of the perturbations, thus making defenses realistic and practical. However, graph node features or language word embeddings do not have an established distance threshold for imperceptibility, which makes the selection of ϵ highly heuristic. Note that, although the perturbation is no longer bounded by an explicit ϵ in FLAG, it is still implicitly bounded in the furthest distance that $\boldsymbol{\delta}$ can reach, i.e. the step size α times the number of ascending steps M.

Also unless otherwise stated, all of the baseline test statistics come from the official OGB leaderboard website, and we conduct all of our experiments using publicly re-

Backbone	ogbn-products Test Acc	ogbn-proteins Test ROC-AUC	ogbn-arxiv Test Acc
GCN	-	**72.51±0.35**	71.74±0.29
+FLAG	-	71.71±0.50	**72.04±0.20**
GraphSAGE	78.70±0.36	**77.68 ±0.20**	71.49±0.27
+FLAG	**79.36±0.57**	76.57±0.75	**72.19±0.21**
GAT	79.45±0.59	-	73.65±0.11
+FLAG	**81.76±0.45**	-	**73.71±0.13**
DeeperGCN	80.98±0.20	85.80±0.17	71.92±0.16
+FLAG	**81.93±0.31**	**85.96±0.27**	**72.14±0.19**

Table 1. Node property prediction test performance on ogbn-products, ogbn-proteins, and ogbn-arxiv datasets. Blank denotes no statistics on the leaderboard.

leased implementations without touching the original model architecture or training setup for fair comparisons. We report mean and standard deviations from 10 runs with different random seeds. Following common practice on this benchmark, we report the test performance associated with the best validation result. We choose GCN, GraphSAGE, GAT, and GIN as our baseline models. In addition, we apply FLAG to the DeeperGCN model to demonstrate its effectiveness on the GNNs with significantly deeper depth. Our implementation always uses $M = 3$ ascent steps for simplicity. Following [15, 26], we use sign$(\cdot)$ for gradient normalization.

Large-scale Node Property Prediction. We summarize the results of node classification in Table 1. Notably, FLAG yields a 2.31% test accuracy lift for GAT, making GAT competitive on the ogbn-products dataset. Considering the specialty of not having input node features in ogbn-proteins, we provide detailed discussions on the effect of different node feature constructions in Section 5. ogbn-mag is a heterogeneous network where only "paper" nodes come with node features. We use the neighbor sampling mini-batch algorithm to train R-GCN and report its results in the Table 2. Surprisingly, FLAG can also directly bring nontrivial accuracy improvement without special designs for heterogeneous graphs, which demonstrates its versatility.

Backbone	ogbn-mag Test Acc
R-GCN	46.78±0.67
+FLAG	**47.37±0.48**

Table 2. Test performance on the heterogeneous OGB node property prediction dataset ogbn-mag.

Large-scale Link Property Prediction. We evaluate our method on two OGB link prediction datasets, which are ogbl-ddi and ogbl-collab. The authors of OGB selected Hits@K as the official evaluation metric. We study the performance of FLAG with GCN and GraphSAGE as backbone on this task. We follow the practice of the baselines to train the models in the full-batch manner. Results are reported in Table 3. We highlight that FLAG brings a salient boost to both GCN and GraphSAGE on the ogbl-ddi dataset.

Backbone	ogbl-ddi Hits@20	ogbl-collab Hits@50
GCN	37.07 ±5.07	44.75±1.07
+FLAG	**51.41±3.76**	**46.22±0.81**
GraphSAGE	53.90 ±4.74	48.10 ±0.81
+FLAG	**63.31±6.06**	**48.44±0.40**

Table 3. Link property prediction test performance on ogbl-ddi and ogbl-collab datasets.

Large-scale Graph Property Prediction. Table 4 summarizes the test scores of GCN, GIN, and DeeperGCN on all four OGB graph property prediction datasets. "Virtual" means the model is augmented with virtual nodes [13, 17, 25]. As adversarial perturbations are crafted by gradient ascent, it would be unnatural and suboptimal to add noises to discrete input node features [45]. We firstly project discrete node features into the continuous space and then adversarially augment the hidden embeddings. On ogbg-molhiv, FLAG yields notable improvements, but when GCN has already been hurt by virtual nodes, FLAG appears to exaggerate the harm. On ogbg-molpcba, GIN-Virtual with FLAG receives an absolute value 1.31% test AP value increase. Besides node classification and link prediction, FLAG's strong effects on graph classification prove its high versatility.

5. Ablation Studies and Discussions

Compatibility with graph structure regularizers. As our augmentation manipulates the input features, it is highly complementary to structure-based regularizers. We validate this point through the experiments below. We mainly focus on two widely-used topological augmentation methods to illustrate [1]: (i) Neighbor sampling [16] randomly samples neighbors for information aggregation. It not only contributes to GNN scalability but also acts as a structure regularizer. A full-batch GraphSAGE reaches $78.50 \pm 0.14\%$ test accuracy on ogbn-products, and neighbor sampling alone generalizes the model to $78.70 \pm 0.36\%$. When FLAG is also used, the test accuracy is increased to $79.36 \pm 0.57\%$. (ii) Virtual node [13] adds one synthetic node that connects to all existing nodes. Nearly all the numbers from Table 4 supports that our method works well with virtual node to generalize GNNs further. Here We highlight one representative group of experiments on ogbg-ppa with GIN as baseline. Vanilla GIN gets $68.92 \pm 1.00\%$ test accuracy. By adding virtual node alone, it goes to $70.37 \pm 1.07\%$. When FLAG is further deployed, test accuracy reaches

[1] We also tried DropEdge [30] but it failed to yield performance gain in the first place.

Backbone	ogbg-molhiv Test ROC-AUC	ogbg-molpcba Test AP	ogbg-ppa Test Acc	ogbg-code Test F1
GCN	76.06±0.97	20.20±0.24	68.39±0.34	31.63±0.18
+FLAG	**76.83±1.02**	**21.16±0.17**	68.38±0.47	**32.09±0.19**
GCN-Virtual	**75.99±1.19**	24.24±0.34	68.57±0.61	32.63±0.13
+FLAG	75.45±1.58	**24.83±0.37**	**69.44±0.52**	**33.16±0.25**
GIN	75.58±1.40	22.66±0.28	68.92±1.00	31.63±0.20
+FLAG	**76.54±1.14**	**23.95±0.40**	**69.05±0.92**	**32.41±0.40**
GIN-Virtual	77.07±1.49	27.03±0.23	70.37±1.07	32.04±0.18
+FLAG	**77.48±0.96**	**28.34±0.38**	**72.45±1.14**	**32.96±0.36**
DeeperGCN	78.58±1.17	27.81$^\natural$±0.38	77.12±0.71	-
+FLAG	**79.42±1.20**	**28.42**$^\natural$**±0.43**	**77.52±0.69**	-

Table 4. Graph property test performance on ogbg-molhiv, ogbg-molpcba, ogbg-ppa, and ogbg-code datasets. ♮ denotes the existence of virtual nodes; blank denotes no statistics on the leaderboard.

Method	GCN	GraphSAGE
w/o BN	71.09±0.22	69.58±0.76
w/ BN	71.74±0.29	71.49±0.27
w/ BN +FLAG	72.04±0.20	72.19±0.21
w/ Dual BN +FLAG	**72.11±0.23**	**72.21±0.20**

Table 5. Test Accuracy on the ogbn-arxiv dataset with different BN methods.

	ogbn-products Test Acc	ogbl-ddi Hits@20	ogbg-molhiv Test ROC-AUC
Baseline	79.45±0.59	53.90±4.74	75.58±1.40
+PGD	80.96±0.41	62.02±6.56	76.14±1.62
+“Free”	79.42±0.84	58.61±6.0	74.93±1.29
+FLAG	**81.76±0.45**	**63.31±6.06**	**76.54±1.14**
+FLAG (fast)	80.64±0.74	-	-

Table 6. Test performances on different datasets trained with different adversarial augmentations. Baselines are GAT, GraphSAGE, and GIN respectively. FLAG (fast) means the training epoch number is decreased to make our method trained as fast as the baseline.

$72.45 \pm 1.14\%$.

Compatibility with batch norm. Batch norm is appearing more and more frequently in top-performing GNNs. [40] argued that there was a potential risk, that adversarial examples could distort BN parameters away from natural distribution so to cause the adversarially trained model to fail on clean samples. The authors proposed to use dual batch norms (one for adversaries and the other for clean ones) at training time to better exploit the generalization ability of adversarial augmentations. To test the dual batch norm method on graph data, we run experiments as summarized in Table 5. We find that the utilization of dual BN can produce a slight performance gain. As there is growing attention on using batch norms on GNNs, it will be interesting to see how to better synergize adversarial augmentation with batch norms in future research.

Comparison with other robust optimization methods. Table 6 shows performances with different adversarial augmentations. For PGD and “free”, we compute 8 ascent steps for the inner-maximization to make the attack strong enough, while for FLAG we only compute 3 steps. We can see that FLAG outperforms all other methods. We attribute that to the practice of our multi-scale augmentation, which diversifies the scale range of feature perturbations, and helps the model see diverse input features to generalize better, especially on out-distribution samples. Although “free” method incorporates diversifying augmentations, but here the benefits are overwhelmed by the suboptimal problem.

Backbone	Test Acc
GAT w/o dropout	75.67±0.27
GAT w/ dropout	79.45±0.59
GAT w/ dropout +FLAG	**81.76±0.45**

Table 7. Test Accuracy on the ogbn-products dataset.

Backbone	ogbn-products Test Acc
GraphSAGE w/ NS	78.70±0.36
+FLAG	**79.36±0.57**
GraphSAGE w/ Cluster	**78.97±0.33**
+FLAG	78.60±0.27
GraphSAGE w/ SAINT	79.08±0.24
+FLAG	**79.60±0.19**

Table 8. Test accuracy on ogbn-products with GraphSAGE trained with diverse mini-batch algorithms.

Effects of weighted perturbation. The effects of biased perturbation are reported in Figure 3c. Generally speaking, when $\log_2(\alpha_u/\alpha_l) > 0$, which means that unlabeled nodes receive larger augmentations, the performance gains are more salient. The phenomenon supports our practice of using weighted perturbation to promote multi-scale augmentations. Empirically we find that the benefit of weighted perturbation is more evident on ogbn-products than on ogbn-arxiv. Our understanding is that, ogbn-products is better suited with our practice of labeled vs. unlabeled split because of its high label sparsity compared with ogbn-arxiv (label rate 8% vs. 54%). When labeled nodes are more sparse, the neighborhood of labeled nodes will be more overwhelmed by unlabeled ones, where our approximation is more accurate.

Hyperparameter sensitivity. Figure 3a and Figure 3b show the hyperparameter sensitivity of our method. Overall, our method is stable to yield consistent accuracy boost compared with baseline.

Compatibility with mini-batch methods. Graph mini-batch algorithms are critical to training GNNs on large-scale datasets. We test how different algorithms will work with adversarial data augmentation with GraphSAGE as the backbone. From Table 8, we see that neighbor sampling [16] and GraphSAINT [43] can all work with FLAG to further boost performance, while Cluster [4] suffers an accuracy drop.

Compatibility with dropout. Dropout is widely used in GNNs. Table 7 shows that, when trained without dropout, GAT accuracy drops steeply by a large margin. What is more, FLAG can further generalize GNN models together with dropout, similar to the phenomenon of image augmentations. It demonstrates that our method is fully compatible with this domain/model-agnostic regularizer.

Towards going "free". FLAG introduces tractable extra training overhead. We empirically show that, when we decrease the total number of training epochs to make it as fast as the standard GNN training pipeline, FLAG still brings significant performance gains. Table 6 shows that FLAG with fewer epochs still generalizes the baseline. Empirically, on a single Nvidia RTX 2080Ti, 100-epoch vanilla GAT takes 88 mins, while FLAG (fast) in Table 6 takes 91 mins. We note that heuristics like early stopping and cyclic learning rates can further accelerate the adversarial training process [39], so there are abundant opportunities for further research on adversarial augmentation at lower or even no cost.

Towards going deep. Over-smoothing stops GNNs from going deep. FLAG shows its ability to boost both shallow and deep baselines, e.g., GCN and DeeperGCN. We carefully examine FLAG's effects on generalization when a GNN goes progressively deeper in Figure 4a. The experiments are conducted on ogbn-arxiv with GraphSAGE as the backbone, where a consistent improvement is evident.

What if there's no node feature? One natural question can be raised: what if no input node features are provided? ogbn-proteins is a dataset without input node features. [17] proposed to average incoming edge features to obtain initial node features, while [24] used summation and achieved competitive results. Note that the GCN and GraphSAGE baselines in Table 1 use the "mean" node features as input and suffer an accuracy drop with FLAG; DeeperGCN leverages the "sum" and gets further improved. Interestingly, when DeeperGCN is trained with "mean" node features, it receives high invariance, so that even large magnitude perturbations will not change its result. The diverse behavior of adversarial augmentation implies the importance of node feature construction method selection.

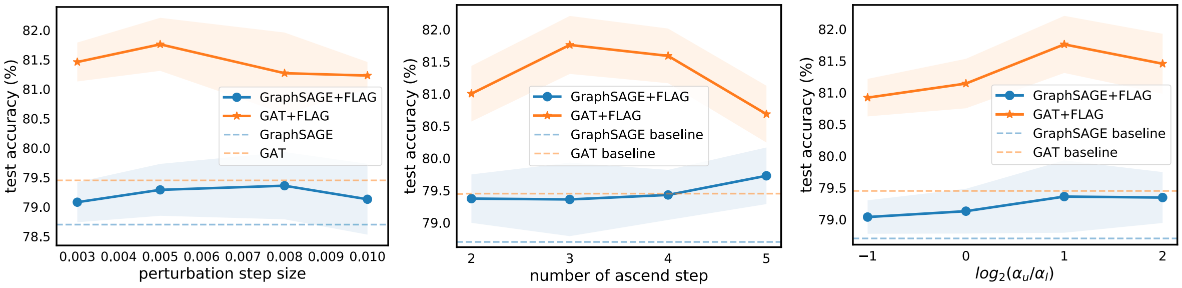
Where Does the Boost Come from? It is now widely believed that model robustness appears to be at odds with clean accuracy. Despite the recent proliferation of literature in using adversarial data augmentation to promote standard performance, it is still unsettled where the boost or detriment of adversarial training comes from. Like one-hot word embeddings for language models, input node features usually come from discrete spaces, e.g., the bag-of-words binary features in ogbn-products. We conjecture that the diverse effects of adversarial training in different domains stem from differences in the input data distribution rather than model architectures. To ground our claim, we have the following observations.

Observation 1: We utilize FLAG to augment MLPs (an architecture where adversarial training has adverse effects in the image domain), and successfully boost generalization. FLAG directly improves the test accuracy from $61.06 \pm 0.08\%$ to $62.41 \pm 0.16\%$ on ogbn-product, and from $55.50 \pm 0.23\%$ to $56.02 \pm 0.19\%$ on ogbn-arxiv.

Observation 2: In general, adversarial training hurts the clean accuracy in image classification, but [35] showed that CNNs could benefit from adversarial augmentations on MNIST, where the pixel values are closer to discrete distribution than other more natural image datasets.

Observation 3: To illustrate, we provide a simple example on the Cora [12] dataset. To simplify the scenario, we choose FGSM to craft adversarial augmentations for a GCN. By adding Gaussian noise with standard deviation σ, we simulate node features drawn from a continuous distribution. The result is summarized in Figure 4b. When $\sigma = 0$, the discrete distribution of node features persists. At this moment, a GCN with adversarial augmentation outperforms the non-augmented model. With increased noise level σ, the features are continuously distributed with large support and FGSM starts to harm the clean accuracy, which validates our conjecture. All these observations support our conjecture that data distribution has more to do with the effect of adversarial augmentation, while the lack of rigorous theoretical justification is a limitation of our analysis.

Applicability to computer vision tasks. Despite the focus on graph learning, we believe our work benefits the vision community. Graph is widely used in CV, e.g., 3D vision and scene understanding. Also 2D images can be represented as grid graphs with pixels as nodes, so we can use GNNs for image recognition smoothly. Here we provide some preliminary results of FLAG on MNIST superpixel dataset [9]. GCN reaches $87.83 \pm 0.70\%$ while GCN+FLAG gets $89.1 \pm 0.37\%$, which is an evidence of FLAG's potential of contributing to the vision community.

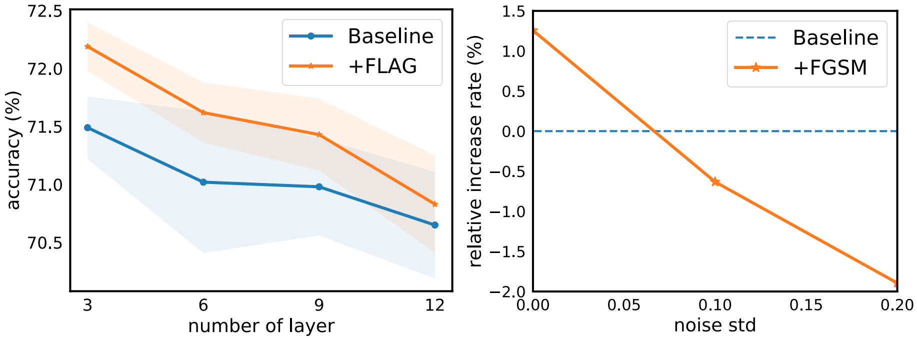


(a) step size

(b) ascent steps

(c) weighted perturbation

Figure 3. Results of GraphSAGE and GAT on the `ogbn-products` dataset.



(a)

(b)

Figure 4. (a) Test accuracy on `ogbn-arxiv`; (b) Performance gap on `Cora`.

6. Conclusion

We propose FLAG, a simple, scalable, and general data augmentation method for better GNN generalization. Like widely-used image augmentations, FLAG can be easily incorporated into any GNN training pipeline. FLAG yields improvements over a range of GNN baselines. Besides extensive experiments, we also provide conceptual analysis to validate adversarial augmentation's different behavior on varied data types. The effects of adversarial augmentation on generalization are still not entirely understood, and we think this is a fertile space for future exploration. However, for the potential negative social impact, our work may be deployed as regularizer of fine-grained social tracker for large-scale social network to undermine personal privacy.

Acknowledgements. Kezhi Kong and Tom Goldstein were supported by DARPA GARD, Office of Naval Research, AFOSR MURI program, the DARPA Young Faculty Award, and the National Science Foundation Division of Mathematical Sciences. Additional support was provided by Capital One Bank and JP Morgan Chase. Guohao Li and Bernard Ghanem were supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding.

References

[1] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019. 1

[2] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1, 3

[4] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019. 7

[5] Zhijie Deng, Yinpeng Dong, and Jun Zhu. Batch virtual adversarial training for graph convolutional networks. *arXiv preprint arXiv:1902.09192*, 2019. 2

[6] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020. 4

[7] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for

graph classification. *arXiv preprint arXiv:1912.09893*, 2019. 4

[8] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 2019. 2

[9] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018. 7

[10] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. *arXiv preprint arXiv:2006.06195*, 2020. 1, 3

[11] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017. 1

[12] Lise Getoor. Link-based classification. In *Advanced methods for knowledge discovery from complex data*, pages 189–207. Springer, 2005. 7

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017. 1, 5

[14] Jonathan Godwin, Michael Schaarschmidt, Alexander L Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and Peter Battaglia. Simple GNN regularisation for 3d molecular property prediction and beyond. In *International Conference on Learning Representations*, 2022. 1

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 3, 5

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017. 1, 2, 5, 7

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020. 1, 2, 3, 4, 5, 7

[18] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019. 2

[19] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*, 2019. 1

[20] Hongwei Jin and Xinhua Zhang. Latent adversarial training of graph convolution networks. In *ICML Workshop on Learning and Reasoning with Graph-Structured Representations*, 2019. 2

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[23] Chang Li and Dan Goldwasser. Encoding social information with graph convolutional networks forpolitical perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604, 2019. 1

[24] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020. 7

[25] Junying Li, Deng Cai, and Xiaofei He. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017. 5

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1, 3, 5

[27] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016. 1

[28] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. 3

[29] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2110–2119, 2018. 1

[30] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019. 1, 2, 5

[31] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3358–3369, 2019. 2, 3

[32] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018. 4

[33] Yantao Shen, Hongsheng Li, Shuai Yi, Dapeng Chen, and Xiaogang Wang. Person re-identification with deep similarity-guided graph neural network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 486–504, 2018. 1

[34] Manli Shu, Zuxuan Wu, Micah Goldblum, and Tom Goldstein. Prepare for the worst: Generalizing across domain shifts with adversarial batch normalization. *arXiv e-prints*, pages arXiv–2009, 2020. 1

[35] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018. 1, 3, 7

[36] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in neural information processing systems*, pages 5334–5344, 2018. 1, 3, 4

[37] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–217, 2020. 1

[38] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019. 1

[39] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 7

[40] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020. 1, 6

[41] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018. 1

[42] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[43] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019. 7

[44] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Thomas Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for language understanding. *arXiv preprint arXiv:1909.11764*, 2019. 1

[45] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018. 5