FLAG: Adversarial Data Augmentation for Graph Neural Networks

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, [2]KAUST, [3]US Naval Academy

The Learning on Graphs and Geometry Reading Group (LoGaG) Presentation
2021.09.28

Background

- Data Augmentation is important to generalization
  - For graphs, we have structural-based augmentations:
    - Neighbor Sampling, DropEdge, Virtual Node, Edge Permutation…
- Question: feature-based augmentations for graphs?

Background

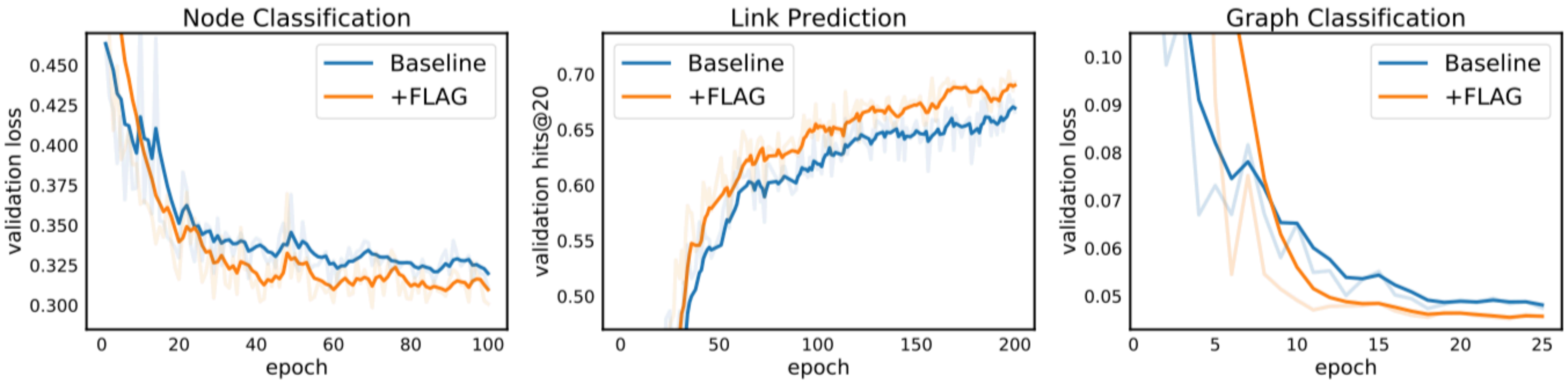


Figure 1: Generalization performance of FLAG on all three tasks. Left: node classification with GAT as baseline on `ogbn-products`; Middle: link prediction with hits@20 as metric (the higher the better) and GraphSAGE as baseline on `ogbl-ddi`; Right: graph classification with GIN as baseline on `ogbg-molhiv`.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

Adversarial Training Is the Cure!

Contributions

- Our work is the first general-purpose feature-based data augmentation method on graph data
- The method works on all the three major graph tasks (node, link, and graph)
- The method has good scalability and works on large-scale datasets
- The method is easy to implement and use
- The method is complementary to existing regularizers (Dropout) and graph structure augmentations (Neighbor sampling & Virtual node)

Method: FLAG

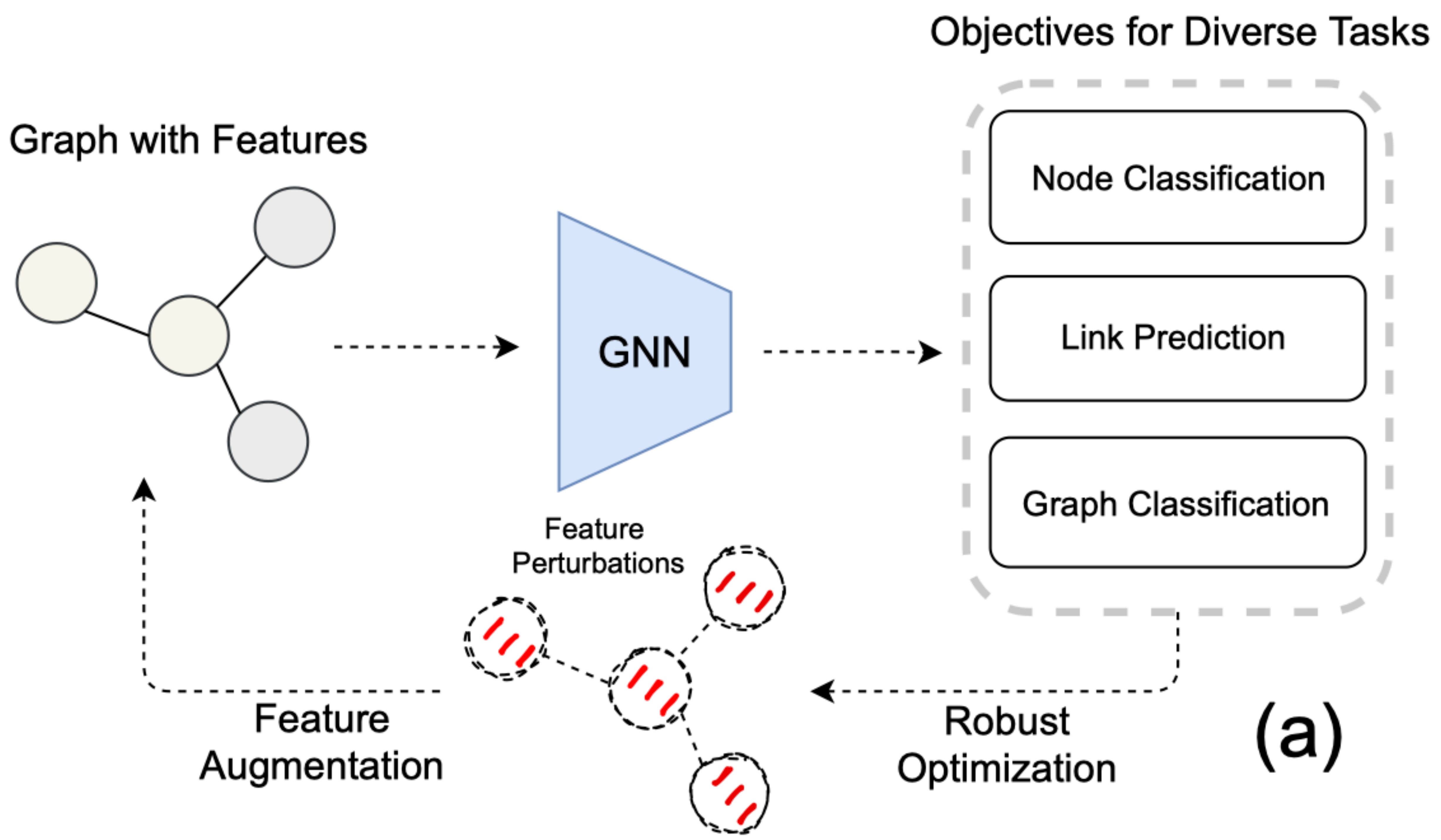
Min-Max Optimization

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

Inner Gradient Ascent

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

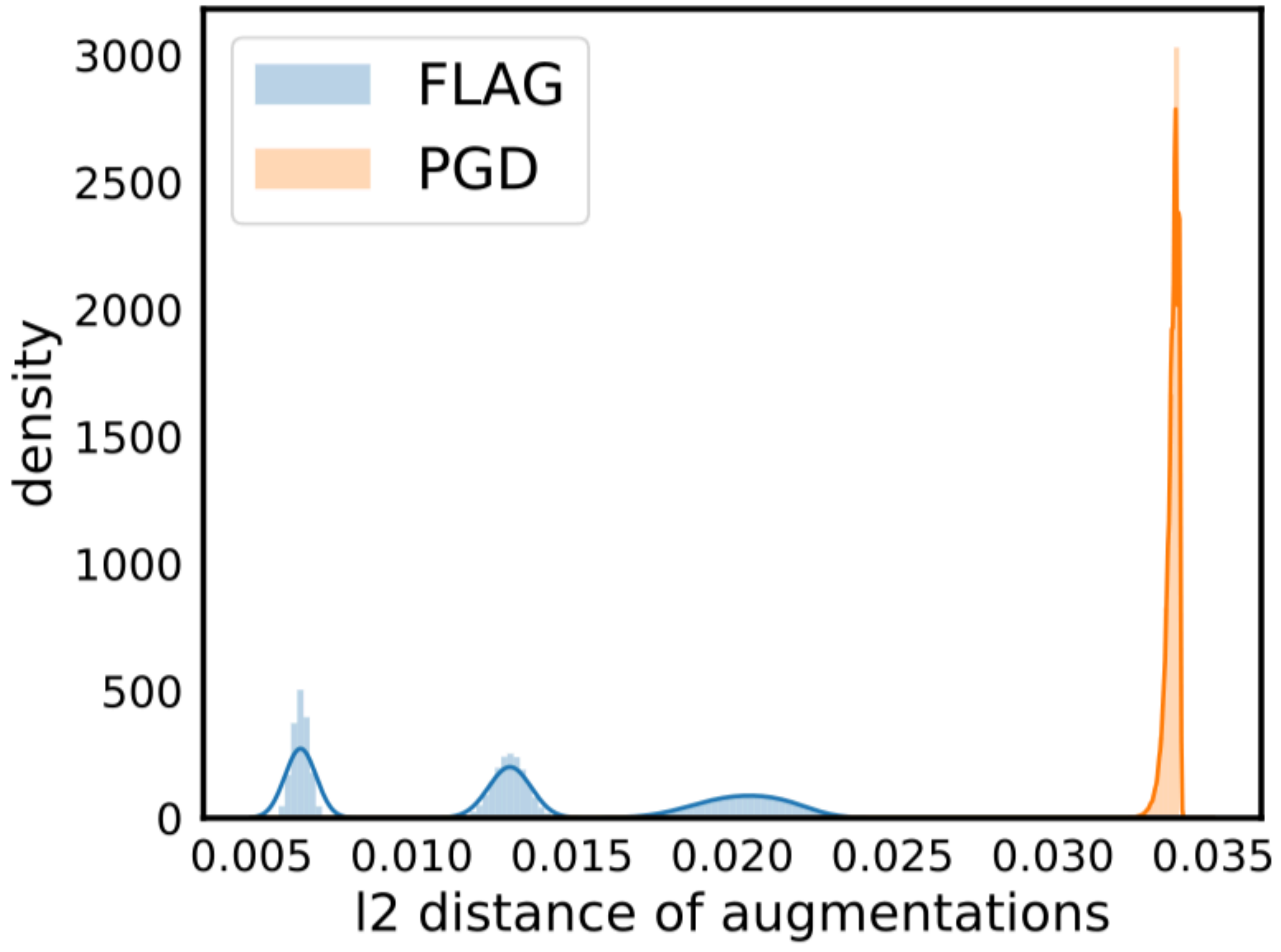
Pipeline



Method: FLAG

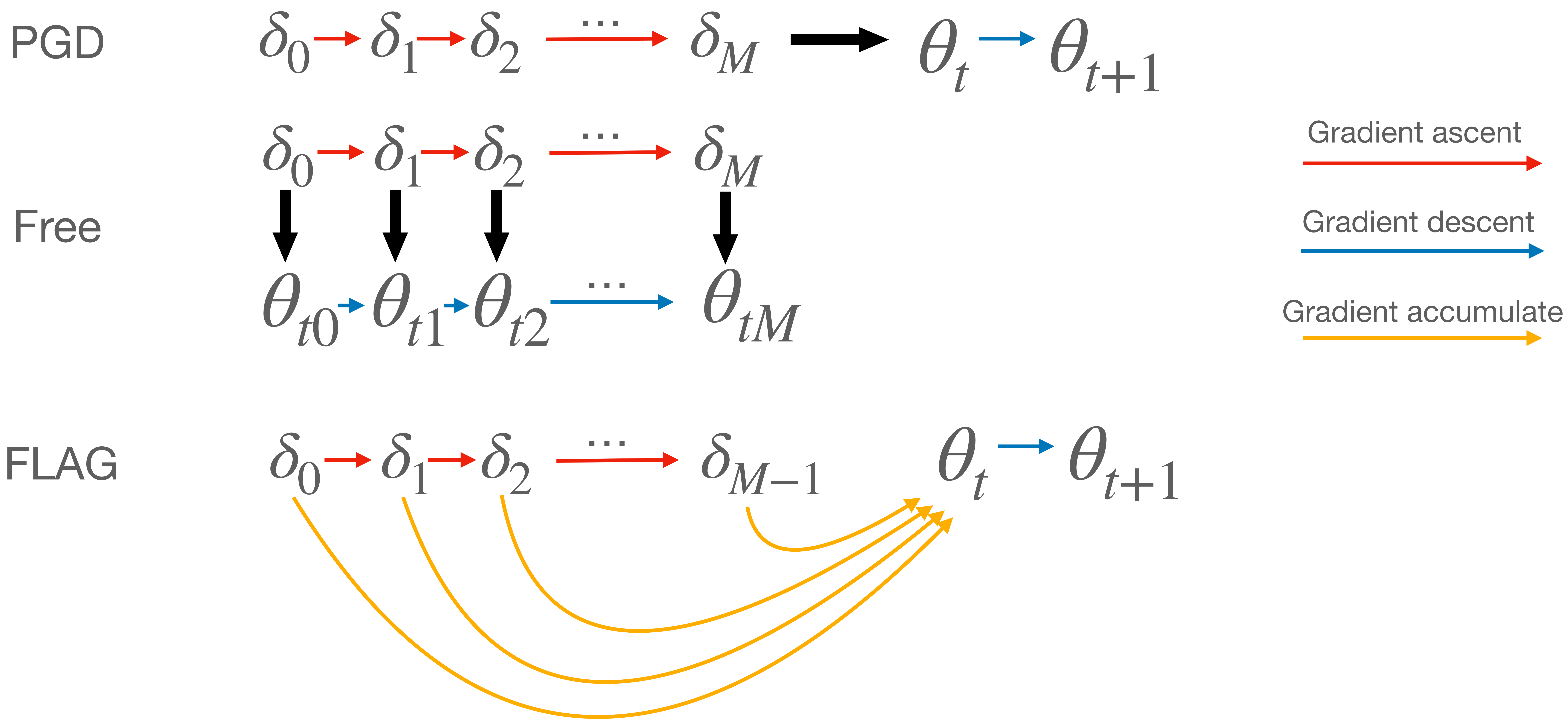
Principle: Multi-scale Augmentation

- "Free" Training
  - Leverage the batch replay technique but improve with gradient accumulation
- Weighted Perturbation
  - For node classification, augment labeled vs. unlabeled with diverse magnitudes (typically larger for unlabeled ones)



Method: FLAG

Principle: Multi-scale Augmentation



Method: FLAG

```
#M as ascent steps, alpha as ascent step size
#X denotes input node features, y denotes labels
def flag(gnn, X, y, optimizer, criterion, M, alpha) :
    gnn.train()
    optimizer.zero_grad()

    pert = torch.FloatTensor(*X.shape).uniform_(-alpha, alpha)
    pert.requires_grad_()
    out = gnn(X+pert)
    loss = criterion(out, y)/M

    for _ in range(M-1):
        loss.backward()
        pert_data = pert.detach() + alpha*torch.sign(pert.grad.detach())
        pert.data = pert_data.data
        pert.grad[:] = 0
        out = gnn(X+pert)
        loss = criterion(out, y)/M

    loss.backward()
    optimizer.step()
```

Experiments

Node Classification

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

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

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

Experiments

Link Prediction

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

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

Experiments

Graph Classification

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.

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♮±0.38	77.12±0.71	-
+FLAG	**79.42±1.20**	**28.42♮±0.43**	**77.52±0.69**	-

Ablation Studies

Table 6: Test accuracy of GAT on `ogbn-products` trained with different adversarial augmentations. FLAG (fast) means the training epoch number is decreased to make our method trained as fast as the baseline.

Backbone	Test Acc
GAT	79.45±0.59
GAT+PGD	80.96±0.41
GAT+"Free"	79.42±0.84
GAT+FLAG	**81.76±0.45**
GAT+FLAG (fast)	80.64±0.74

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

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**

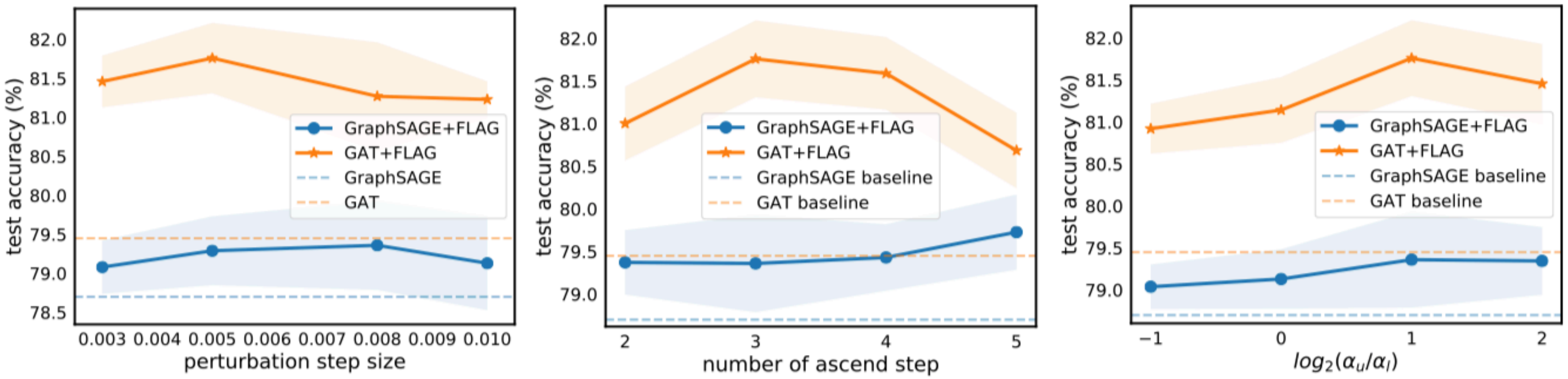
Table 7: Test Accuracy on the `ogbn-products` dataset.

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**

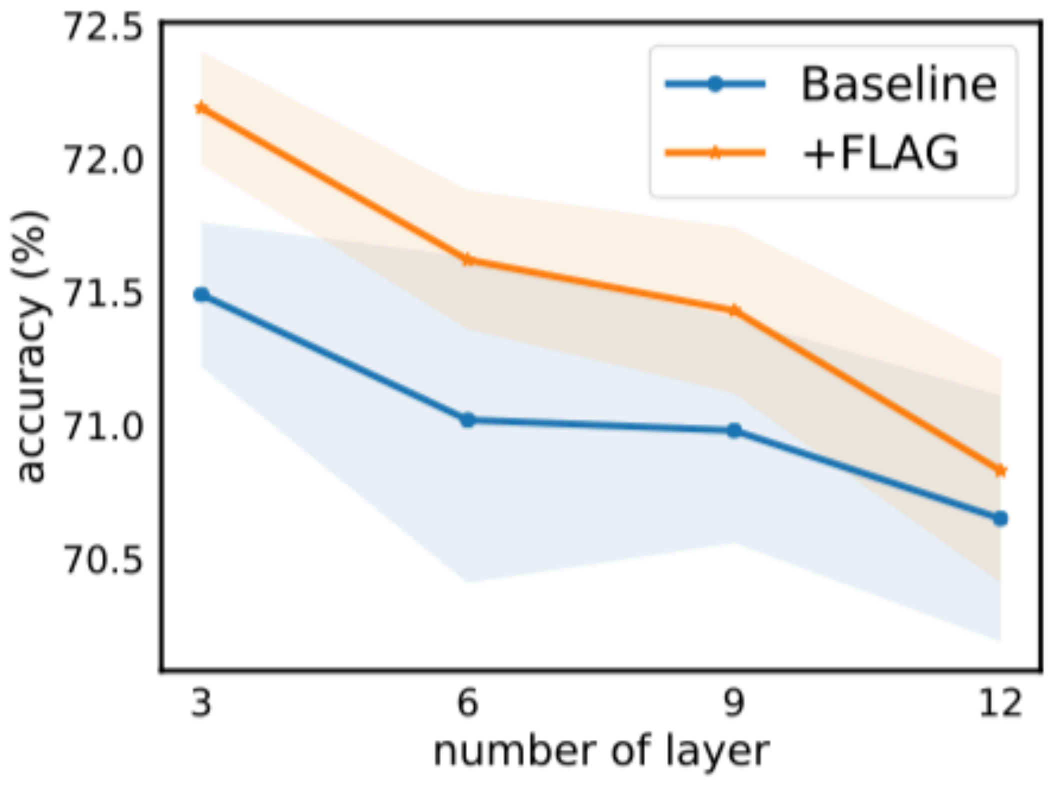
Table 8: Test accuracy on `ogbn-products` with GraphSAGE trained with diverse mini-batch algorithms.

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**

Ablation Studies



(a) step size (b) ascent steps (c) weighted perturbation



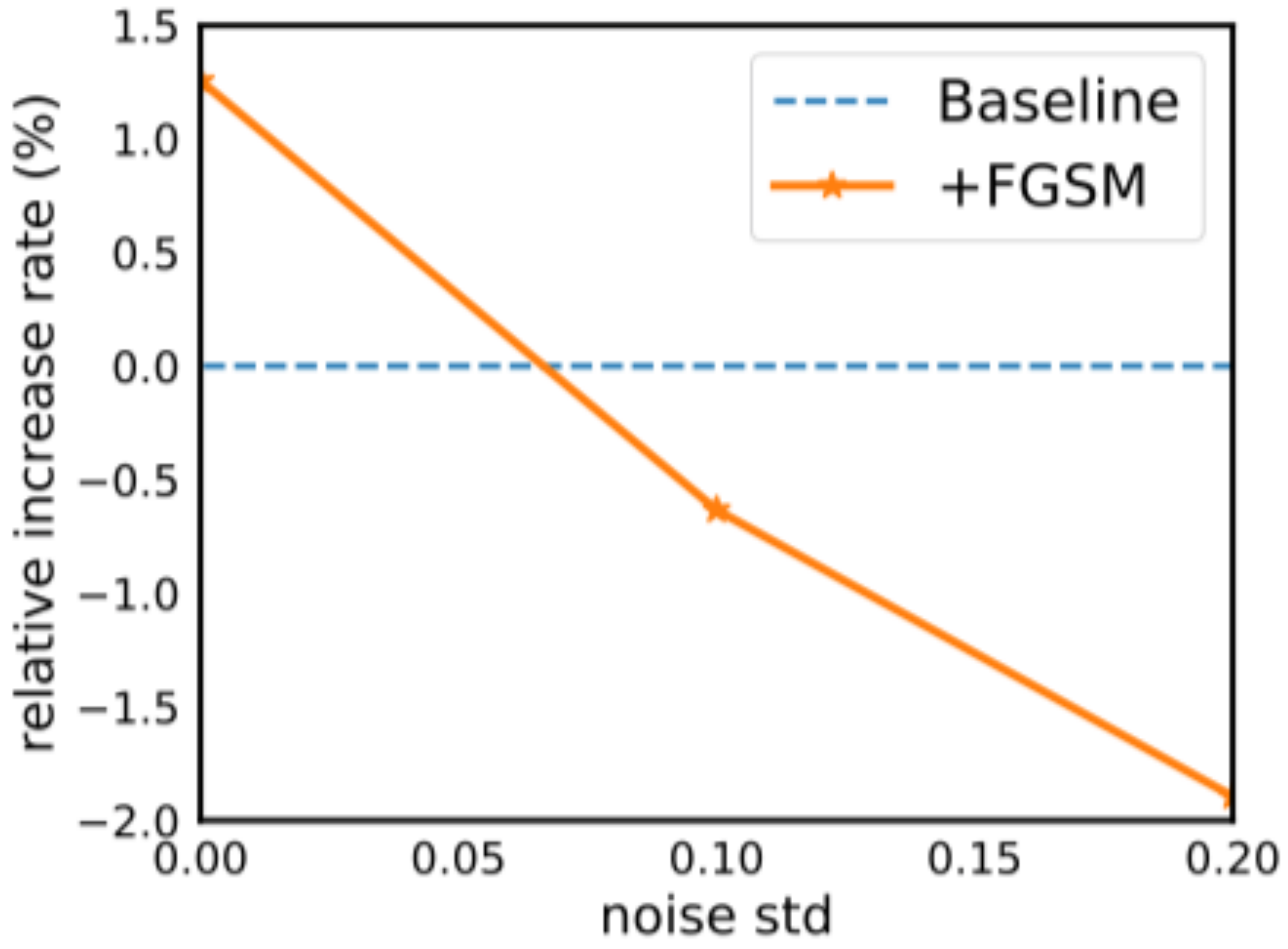
(a) model depth

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

Analysis

- Guess: Discrete features are the ones that should be adversarially augmented
- Observations
    1. FLAG boosts MLPs on graph datasets
        A. 61.06±0.08% to 62.41±0.16% on ogbn-product
        B. 55.50 ± 0.23% to 56.02 ± 0.19% on ogbn-arxiv
    2. Adversarial Training boosts performance of CNNs on MNIST [Tsipras et al. 2018]
    3. Gaussian noises destroy the boost

Cora toy example



(b) noise std

Tsipras, Dimitris, et al. "Robustness May Be at Odds with Accuracy." International Conference on Learning Representations. 2018.

Limitations

- FLAG is empirically oriented, which lacks theoretical motivation
- FLAG introduces time overhead
- FLAG may not work when there are no initial node features

Thanks for listening!

- Paper: https://arxiv.org/abs/2010.09891
- Code: https://github.com/devnkong/FLAG