Response to the Reviewers

We thank the reviewers for their critical assessment and insightful comments on our work. We have made extensive modifications to our manuscript. In the following we address their concerns point by point.

Summary of Changes

TODO

Reviewer 1

Reviewer Point P 1.1 — The authors conduct an empirical analysis of performance bottlenecks in graph neural network training. The authors identify the edge-related calculation is the performance bottleneck. Experimental of several GNN variants, such as GCN, GGNNNN, GAT and GaAN on six real-world graph datasets verify the importance of the findings. The experimental analysis is sufficient. However, there are some tiny issues in this paper.

Reply: Thank you for your kindly comments on our manuscript. We have revised the manuscript according to your suggestions. Please see the replies below.

Reviewer Point P 1.2 — There are lots of symbols in this paper. Some symbols are reused and confusing, such as s denotes sub-layers or edge features.

Reply: We apologize for the confusing use of symbols. To clarify the symbol usage, we have unified the usage of symbols in the revised manuscript. After revision, each symbol only represents one meaning.

Specifically, we use $\boldsymbol{s}$ to represent aggregated vectors in graph neurons. We use $\boldsymbol{s}_x^l$ to denote the aggregated vector of the graph neuron corresponding to the vertex v_x in the GNN layer l. If the GNN layer l has sub-layers, $\boldsymbol{s}_x^{l,i}$ represents the aggregated vector of v_x in the i-th sub-layer. As for edges and their feature vectors, we use $e_{y,x}$ to represent an edge pointing from v_y to v_x and we use $\boldsymbol{e}_{y,x}$ to represent its feature vector.

In order to facilitate readers to quickly locate the meaning of symbols, we have added Table 1 at the beginning of Section 2 "Review of Graph Neural Networks" in the revised manuscript.

Table 1: Frequently-used Symbols

Category	Symbol	Meaning
Graph Structure	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The simple undirected input graph with the vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$.
	v_x	The x-th vertex of the input graph.
	$e_{y,x}$	The edge pointing from v_y to v_x of the input graph.
	$\mathcal{N}(v_x)$	The adjacency set of v_x in the input graph.
	$\bar{d}$	The average degree of the input graph.
GNN Definition	L	The number of GNN layers.
	K	The number of heads in a GNN layer.
	ϕ^l	The messaging function of the GNN layer l.
	Σ^l	The aggregation function of the GNN layer l.
	γ^l	The vertex updating function of the GNN layer l.
	$\phi^{l,i}$ / $\Sigma^{l,i}$ / $\gamma^{l,i}$	The messaging/aggregation/updating function of the i-th sub-layer of the GNN layer l.
	$\boldsymbol{W}^l, \boldsymbol{W}^{(k)}/\boldsymbol{b}, \boldsymbol{a}$	The matrices/vectors represented by the blue characters are the weight matrices/vectors that need to be learned in the GNN.
Vector	$\boldsymbol{v}_x$	The feature vector of the vertex v_x.
	$\boldsymbol{e}_{y,x}$	The feature vector of the edge $e_{y,x}$.
	$\boldsymbol{h}_x^l$	The input hidden vector of the graph neuron corresponding to v_x in the GNN layer l.
	$\boldsymbol{h}_x^{l+1}$	The output hidden vector of the graph neuron corresponding to v_x in the GNN layer l.
	$\boldsymbol{m}_{y,x}^l$	The message vector of the edge $e_{y,x}$ outputted by ϕ^l of the GNN layer l.
	$\boldsymbol{s}_x^l$	The aggregated vector of the vertex v_x outputted by Σ^l of the GNN layer l.
	$\boldsymbol{h}_x^{l,i}$ / $\boldsymbol{m}_{y,x}^{l,i}$ / $\boldsymbol{s}_x^{l,i}$	The hidden/message/aggregated vector of the vertex v_x outputted by $\gamma^{l,i}$/$\phi^{l,i}$/$\Sigma^{l,i}$ of the i-th sub-layer of the GNN layer l.
	d_{in}^l, d_{out}^l	The dimension of the input/output hidden vectors of the GNN layer l.
	$dim(\boldsymbol{x})$	The dimension of a vector $\boldsymbol{x}$.

Reviewer Point P 1.3 — Some typical applications of GNNs should be included, such as video object segmentation [ref1], human-object interaction [ref2] and human-parsing [ref3].[1] Zero-shot video object segmentation via attentive graph neural networks,iccv 2019 [2] Learning human-object interactions by graph parsing neural networks, eccv 2018. [3] Hierarchical human parsing with typed part-relation reasoning, cvpr 2020.

Reply: Thank you for pointing out missing references. Computer vision is indeed an important application area of graph neural networks. We have added the mentioned references in Section 1 "Introduction" in the revised manuscript.

Reviewer Point P 1.4 — There are some grammar errors and typos:
- 'Take the demo GNN in Figure 1(a) as the example.'
- 'to calculate the output hidden vector h^{l+1} of the current layer l, i.e., $h^{l+1} = \gamma^l(h^l, s^l)$ The end-to-end training requires. . . '
- 'Implementing it with the specially optimized basic operators on the GPU is a potential optimization'
- The sentences in the experimental section should be unified.

Reply:

Thank you for pointing them out. We feel sorry for our carelessness. We have proofread our revised manuscript to eliminate grammar errors and typos. We have also unified the tenses of the sentences in Section 3 "Evaluation Design" and Section 4 "Evaluation Results and Analysis". Specifically, we use the past tense to describe experimental methods, results and what they indicate. We use the present tense in the sentences that FigureX or Table X is the subject of the sentences.

Reviewer Point P 1.5 — Figures 6 and 7 should be adjusted. The figures and fonts are too small.

Reply:

As suggested, we have split Figure 6 and Figure 7 into five figures (Figure 6 to 10) in the revised manuscript to enlarge their subfigures. We have also adjusted font sizes in all figures (besides Figure 6 and Figure 7) to make sure that they are equal to or greater than the font size of figure captions.

Reviewer Point P 1.6 — In my view, computation efficiency is to describe the testing or validation process. Except for reporting and analyzing the training times, it is meaningful to discuss the inference time. This is also an important point of view for deep learning researchers to be concerned about.

Reply: Thank you for the insightful comment and suggestion. The efficiency of inference (including inference time and memory usage) is indeed important for deep learning researchers and engineers. We add brief description on GNN inference in Section 2.5 "Inference with GNNs" in the revised manuscript. As suggested, we also use the same methodology as the training phase to analyze the performance bottlenecks in the inference phase and we add corresponding experimental results in every subsection of Section 4 "Evaluation Results and Analysis".

1. In Section 4.1 "Effects of Hyper-parameters on Performance", we conduct extra experiments on how the hyper-parameters of GNNs affect the time and peak memory usage of inference. We find that the effects of hyper-parameters on the inference time and memory usage are the same as the training time. When the other hyper-parameters are fixed, each hyper-parameter itself affects the training time and memory usage in a linear way. The complexity analysis of GNNs holds for both training and inference. We present specific experimental results in Section 4.1.2 "Effects on Inference" in the revised manuscript.

2. In Section 4.2 "Time Breakdown Analysis", we additionaly conduct time breakdown analysis for GNN inference and compare the similarites and differences between training and inference. As for the similarities, we find that the performance characteristics of inference are highly similar to training on the layer level and the step level of edge calculation. The edge calculation stage is the main performance bottleneck for both training and inference. The specific performance bottlenecks of the edge calculation stage depend on the time complexity of the messaging function ϕ of the GNN. The differences between training and inference were mainly reflected in two aspects: the elapsed time and the top time-consuming basic

operators. The inference time is 34% (GCN), 32% (GGNN), 25% (GAT), and 32% (GaAN) of the training time on average, as shown in Figure 1. Since the inference only conducted the forward propagation from the input layer to the prediction layer, the inference time is very close to the time of the forward phase in training. For the top time-consuming basic operators, some top operators from the backward phase in training are replaced by operators from the prediction layer in inference. For example, Figure 2 shows that the `gather` operator used in the backward phase of training is replaced by the `index` operator used in the prediction layer of GCN. Though the the top time-consuming basic operators are changed, they still indicates that GPUs are suitable for conducting GNN inference. We present more details of time breakdown analysis on GNN inference in Section 4.2.4 "Performance Bottlenecks in Inference" in the revised manuscript.

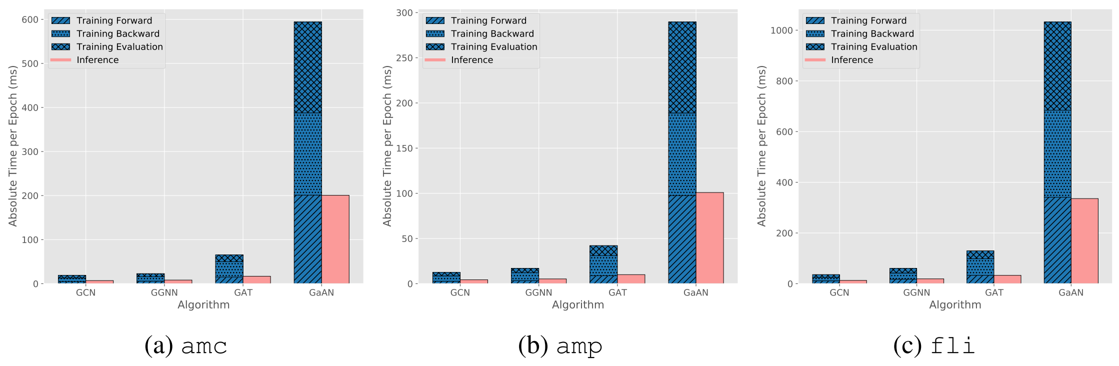


Figure 1: Comparison of wall-clock training time and inference time on different datasets.

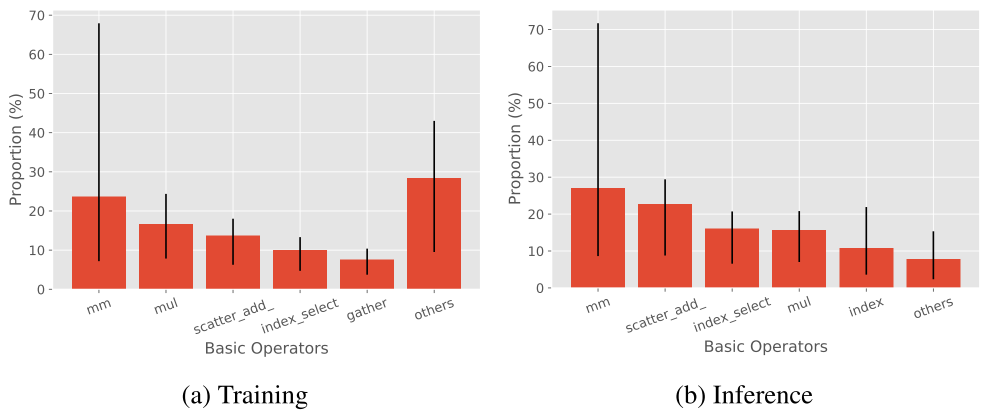
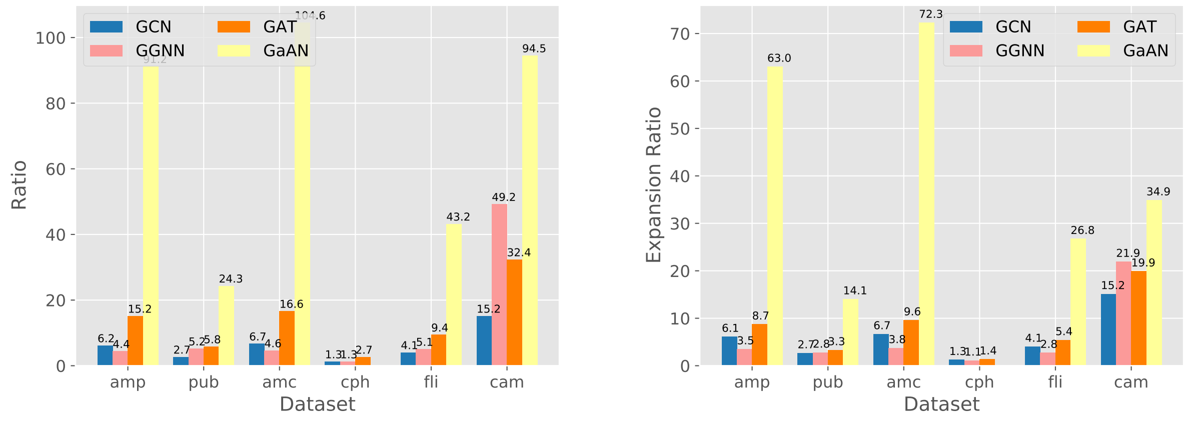


Figure 2: Top 5 time-consuming basic operators of GCN. The time proportion of each basic operator is averaged over all datasets with the error bar indicating the maximum and the minimum.

3. In Section 4.3 "Memory Usage Analysis", we conduct extra experiments to evaluate the peak memory usage during inference. We find that the memory usage of inference also

mainly comes from the edge calculation stage, same as training. Compared with training, the memory expansion ratios of inference are much less, as shown by Figure 3. The memory expansion ratios of inference are 45% to 83% (GGNN), 52% to 61% (GAT), and 37% to 69% (GaAN) of training. GGNN, GAT and GaAN have to cache some intermediate results during training, the memory space of those results is saved during inference. However, the memory expansion ratios of inference are still high, preventing us from inferencing on big graphs. More details are available in Section 4.3.2 "Memory Usage of Inference" in the revised manuscript.



(a) Training (b) Inference

Figure 3: Memory expansion ratios of typical GNNs.

4. In Section 4.4 "Effects of Sampling Techniques on Performance", we additionally conduct experiments to analyze performance bottlenecks in sample-based inference. Under the same batch size, we find that the subgraph sampler used in inference tends to produce larger subgraphs than the neighbor sampler used in training. Thus, the batch size used in inference should be small to avoid the out of memory exception on the GPU side. However, the overheads brought by sampling and data transfering from CPU to GPU become obvious when the batch size was small. They account for near half of the total inference time on the `amp` and `cam` datasets, as shown in Figure 4. The results indicate that the current implementation of inference sampler in PyG is still inefficient. We present more details of our experimental analysis in the "Inference Time" paragraph of Section 4.4.3 "Performance Bottlenecks Analysis"

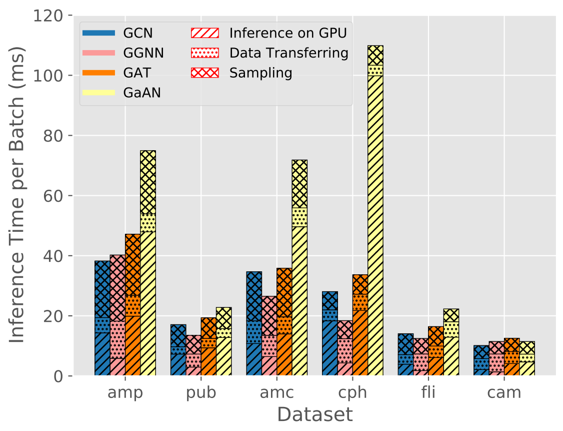


Figure 4: Inference time per batch breakdown under the batch size of 2048.

Reviewer 2

Reviewer Point P 2.1 — In the paper, authors accomplished a unique study and analysis on GNN models training complexity. The articles first review and development history of GNNs and creatively model all architectures as input layers, intermediate layers of graph neurons and prediction layers. And they quantitatively summarize the time and space complexity of 4 representative GNNs, including graph convolution, gated recurrent graph net, graph attention net and GraphSage. Most importantly, the article first break down complexity into operator level and offered analysis of good granularity, giving reader more guidance in future study. At last, the solid experiments included the study of effects of hyper-parameters and a comparison of two major sampling techniques: neighbor sampling and cluster sampling.

Reply: Thank you for your positive comments on our manuscript. We have carefully revised the manuscript according to your kindly comments and suggestions in the following points.

Reviewer Point P 2.2 — In general, the paper was well written and organized with good structure and clear narratives. Just some minor language errors like line Page 8, Line 208, ”In active graph neurons” =>”Inactive graph neurons”.

Reply: We feel sorry for our carelessness. We have proofread our revised manuscript to eliminate such language errors.

Reviewer Point P 2.3 — I was impressed by the way that authors categorize layers and operators in GNNs, very clear and instructive.

It is also pretty neat to divide layer time complexity into two buckets: vertex calculation and edge calculation. The data model pretty well summarizes mainstream GNN layer architectures. And this analysis is very insightful for layer profiling.

And the experimental evaluation were done over 6 large graph-structured datasets.

Reply: Thank you for the positive comment.

Reviewer Point P 2.4 — While, one major drawback is that I did not clearly see the analysis complexity v.s. accuracy. For example, in Figure 19 and 20, I did not see network accuracy from those 4 GNNs. There is always tradeoff between model complexity and model performance, and in some scenarios where high complexity is allowed, a sophisticated model of more powerful representation capability is still needed.

Reply: Thank you for the valuable suggestions. The model complexity directly affects the accuracy. In order to analyze the relationship between model complexity and accuracy, we have conducted two kinds of extra experiments in the revised manuscript: (1) how the hyper-parameters of the GNNs (like the dimension of hidden vectors and the number of heads) affect the accuracy of GNNs (in Section 4.1.4); (2) how the batch size in the sampling methods affects the accuracy of GNNs (in Section 4.4.3). In this reply, we focus on the first kind of experiments. We focus on the second kind of experiments in the next reply.

The hyper-parameters determine the model complexity of GNNs and thus affect accuracy. To find out the effects of hyper-parameters, we measure and compare the test accuracy of the four GNNs trained with different hyper-parameters. We have two main findings.

First, the accuracy of GNNs is much more sensitive to the dimension of hidden vectors (for GCN/GGNN/GaAN) and the dimension of each head (for GAT) than the other hyper-parameters. Figure 5 shows the results of GaAN. More results are available in Section 4.1.4 “Effects on Accuracy” in the revised manuscript. The accuracy of the four GNNs is low when the dimensions are very low (≤ 8). As the dimensions increase to a certain threshold, the GNNs gain sufficient representation capability to achieve good accuracy. But very high dimensions (≥ 2048) are also harmful to accuracy in some cases, as they increase the risk of overfitting.

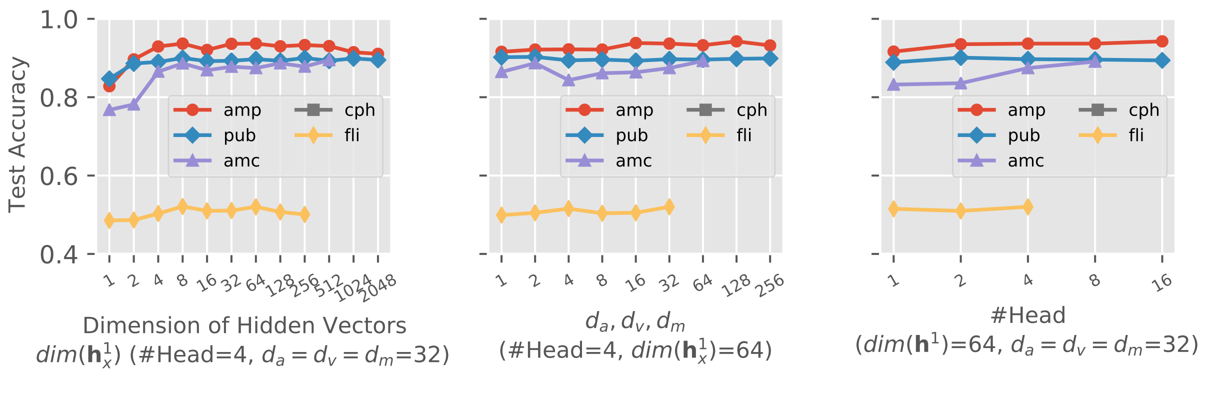


Figure 5: Effects of hyper-parameters on accuracy of GaAN.

Second, the relative accuracy between the four GNNs varies greatly with different datasets. We compare the best accuracy that each GNN achives on every dataset in Figure 6. The best accuracy of the four GNNs are very close. It is also close to the accuracy reported in [Shchur et al. (2018); Zeng et al. (2020)]. There is no clear winner in terms of accuracy. GaAN that has the highest complexity performs slightly better than the other GNNs. Simple GNN models (such as GCN) can still achieve good acuuracy with proper hyper-parameter settings.

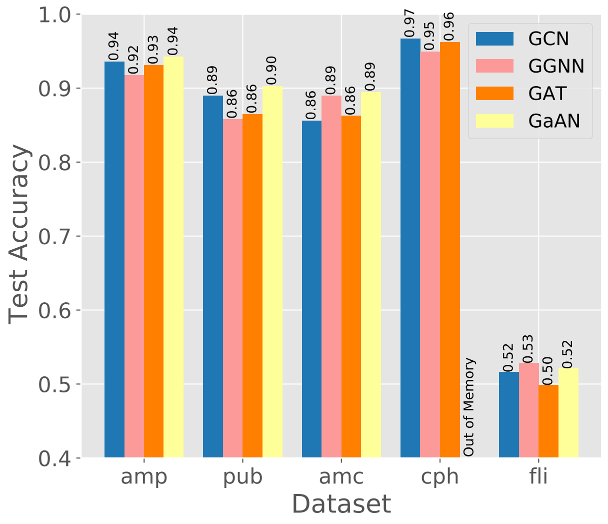


Figure 6: Best accuracy that each GNN achieves on different datasets.

In addition, we add more details on the proportions of the train/evaluation/test sets of the datasets in Section 3.1 "Experimental Setting" in the revised manuscript. Since the vertices of the `cam` dataset are not attached with feature vectors, we exclude it from the accuracy evaluation.

Reviewer Point P 2.5 — Sampling method is definitely going to reduce model complexity, since all models complexity depend on graph node number N, while performance is going to be compromised as well. I would like to see authors resolve the concern of significant accuracy drop after applying aggressive sampling of subgraphs.

Reply:

Thanks for the insightful comment. With or without the sampling methods, the structure of the GNN remains the same, but the gradients of the model parameters are calculated differently, affecting the accuracy of the GNN model. With sampling, each GNN layer still consists of $|\mathcal{V}|$ graph neurons ($|\mathcal{V}|$ is the number of vertices in the graph), but only graph neurons corresponding to the vertices in the sampled subgraph are activated. Thus, the model parameters are updated according to the gradients calculated on the sampled subgraph, instead of the whole graph. In other words, the GNN models are trained in a *batch* gradient descent manner (i.e., full-batch training) without sampling. With sampling, the GNN models are trained with *mini-bach* stochastic gradient descent.

To find out how the sampling techniques affect accuracy, we train the four typical GNNs with different batch sizes and compare the accuracy with the full-batch training. For each combination of dataset and GNN, we use the hyper-parameters that achieve the highest accuracy under the full-batch training. Figure 7 and Figure 8 shows the accuracy comparisons of the two typical sampling methods under different batch sizes. Results of the other datasets are available in Section 4.4.3 "Effects on Accuracy" in the revised manuscript. In most cases, the test accuracy of the sampling methods is just slightly lower than the accuracy of the full-batch training when the relative batch size $\geq 3\%$. In several cases (like GaAN in Figure 7a and Figure 7b), the accuracy of the neighbor sampler was even higher than the full-batch training. Among the two sampling methods, the performance of the cluster sampler is stabler than the neighbor sampler. The test accuracy of the cluster sampler is close to the accuracy of full-batch training, while the test accuracy of the neighbor sampler shows more obvious differences for different GNNs.

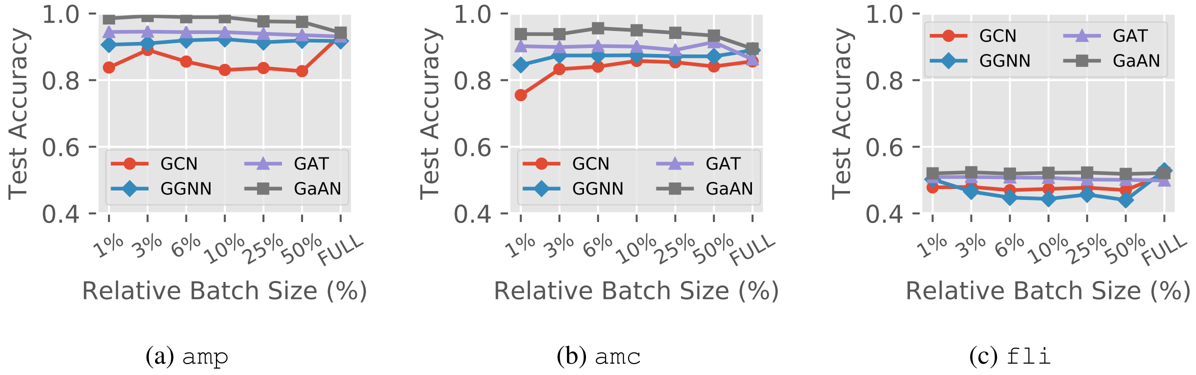


Figure 7: Test accuracy under different batch sizes of the neighbor sampler. FULL means that the full graph participated in the training.

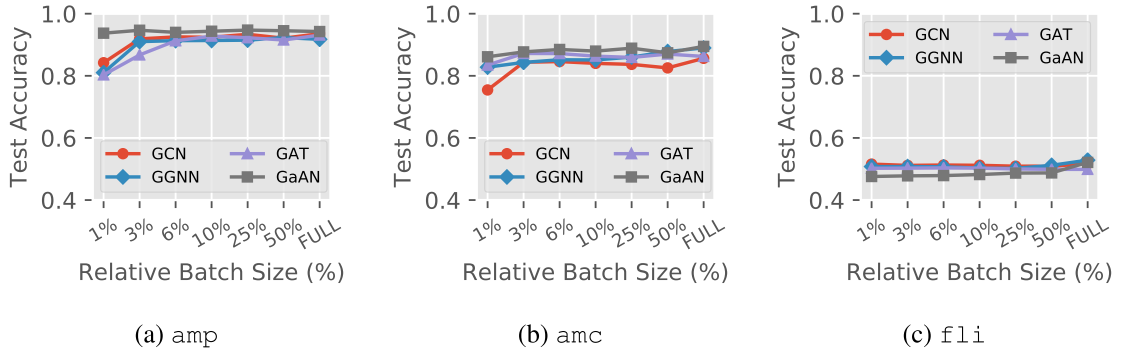


Figure 8: Test accuracy under different batch sizes of the cluster sampler. FULL means that the full graph participated in the training.

The results indicate that the relationships between batch size and test accuracy are complex, especially for the neighbor sampler. A larger batch size does not always bring higher accuracy, like GAT in Figure 7c. Our observations are similar to Zeng et al. (2020). How to automatically select a proper batch size is a topic worth studying.

Reviewer Point P 2.6 — Hope authors supplement the effect of sampling and GNNs on accuracy while comparing different complexity of model and sampling methods.

Reply:

We are grateful for your insightful suggestion. As suggested, we additionally evaluate the effects of model complexity on accuracy in Section 4.1 "Effects of Hyper-parameters on Performance" in the revised manuscript. We add experimental results on how the hyper-parameters of GNNs affect the accuracy. We also compare the test accuracy of different GNN models on the same dataset.

We also additionally evaluate the effects of sampling techniques on accuracy in Section 4.4 "Effects of Sampling Techniques" in the revised manuscript. We presented the experimental results in a new subsection Section 4.4.3 "Effects on Accuracy".

References

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. pages 257—-266. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM.

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. pages 1024–1034. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. K. (2020). Graphsaint: Graph sampling based inductive learning method. Proceedings of the 8th International Conference on Learning Representations (ICLR), OpenReview.net.