We sincerely appreciate all reviewers' valuable feedback and provide our detailed responses below. Here 1BG represents 1BG-S if not specifically stated.

1 To Reviewer 1

1. The theoretical part of Theorem 3 is wrong and misleading.

Sorry for the notation mistakes. Actually, the assumptions of gradients are considered under the quantization situation $\hat{\mathbf{C}}$. It should be $\hat{\boldsymbol{G}}$, not $\boldsymbol{G}$ here. After this correction, the theoretical part is sound. The correct assumptions of Theorem 3 should be:

Under the quantization situation, we have the SGD in the form of $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta\hat{\boldsymbol{G}}$, starting from the initial weights $\boldsymbol{W}_1$. We make the following assumptions:

- The loss $\mathcal{L}$ is continuous differentiable and $\nabla\mathcal{L}(\boldsymbol{W})$ is β-Lipschitz continuous (Lip, 2022).
- $\mathcal{L}$ is bounded below by $\mathcal{L}_{inf}$.
- From Theorem 2 in our paper, we can assume there exists $\sigma^2 > 0$, such that $\forall \boldsymbol{W}, \mathrm{Var}\left[\hat{\boldsymbol{G}}\right] \leq \sigma^2$, where for any vector $\mathbf{x}, \mathrm{Var}[\mathbf{x}] := \mathbb{E}\|\mathbf{x}\|^2 - \|\mathbb{E}[\mathbf{x}]\|^2$.

$\boldsymbol{W}_t = \left\{\boldsymbol{W}^{(l)}\right\}_{l=1}^{L}$ is a flattened vector of the parameters at the t-th iteration, and $\hat{\boldsymbol{G}}_t = \left\{\hat{\boldsymbol{G}}_t^{(l)}\right\}_{l=1}^{L}$ is the corresponding AC gradient. $\mathcal{L}(\boldsymbol{W}_t)$ is the batch loss at the t-th iteration, where $\nabla\mathcal{L}(\boldsymbol{W}_t) = \mathbb{E}\left[\hat{\boldsymbol{G}}_t\right]$. Under quantization, we have the SGD iteration $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta\hat{\boldsymbol{G}}$. Let $\mathbb{E}[\cdot \mid t]$ and $\mathrm{Var}[\cdot \mid t]$ be the expectation and variance taken only over the minibatch and quantizations at the t-th iteration. Then according to the detailed proof in Bottou et al. (2018). We can have the convergence theorem under quantization.

Theorem 3 (Convergence) If $0 < \eta \leq \frac{1}{\beta}$, for iteration t in $\{1, ..., T\}$, where T is the maximum number of iterations, we have

$$\mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{W}_t)\right\|^2 \leq \frac{2\left(\mathcal{L}(\boldsymbol{W}_1) - \mathcal{L}_{inf}\right)}{\eta T} + \eta\beta\sigma^2$$

2. Evaluation of quantization error.

Thanks for your constructive advice! Following your suggestion, we calculate the absolute error as $\|\mathbf{a} - \mathbf{b}\|$ for quantized embeddings and gradients according to the definition. We train GraphSAGE on Reddit (N=4) when bit=1 for 500 epochs. From Fig. 1 (a), we can see the approximation error of embedding is decreasing with epochs on the whole, showing that quantization brings limited (bounded) error to embeddings in GNN training. Though gradient error (Fig. 1 (a)) does not tend to decrease, it remains at a lower level compared with embeddings. Table 1 also shows their average error. Due to time limit, we will put the in-depth analysis in the future work. In summary, extreme 1-bit quantization inevitably brings errors to embeddings and gradients, but GNNs have much higher tolerance for error caused by reduced precision compared to DNNs. And as we stated in Section-3.5, we only quantize part of them so that the compound error can be decreased.

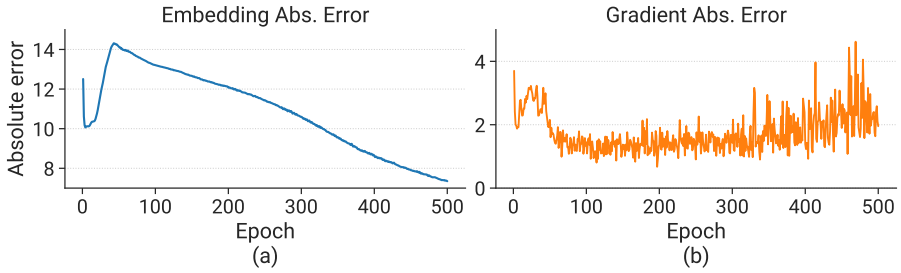

Figure 1. Absolute error of embeddings and gradients.

The jacobian matrix (gradient) between two tensors can be viewed as a sum over all "paths" connecting these two tensors (Bauer, 1974). And the architecture of most GNNs is considerably shallower than current DNNs. Informally, for GNNs, the key intuition is that these paths are significantly "shorter" compared to those of CNNs. As a result, the approximation error (i.e., the variance from quantization) along the path is hard to get accumulated (Liu et al., 2021). Theorem 2 makes the

Embedding error	Gradient error
10.8	1.8

Table 1. Average quantization error.

above statement more rigorous. For the gradients under the quantization strategy, we have

$$\mathrm{Var}[\hat{\boldsymbol{G}}^{(l)}] = \mathrm{Var}\left[\boldsymbol{G}^{(l)}\right] + \sum_{m=l}^{L} \mathbb{E}\left[\mathrm{Var}\left[\mathbf{B}^{(l\sim m)}\left(\hat{\boldsymbol{J}}^{(m)}, \hat{\mathbf{C}}^{(m)}\right) \mid \hat{\boldsymbol{J}}^{(m)}\right]\right]$$

where the term $\mathrm{Var}\left[\mathbf{B}^{(l\sim m)}\left(\cdot, \hat{\mathbf{C}}^{(m)}\right)\right]$ is the variance introduced by utilizing the quantization context. The key insights from it are two folds. First, since the variances introduced by compressed contexts at different layers will accumulate, it suggests that the noise introduced by the compressed context is relatively small for shallow models. Since GNNs usually have fewer layers than popular DNN models, this may explain why the accuracy loss is negligible in our experiments. Second, it tells how much extra variance the quantization introduces. We may reduce the numerical precision to lower bits if the quantization variance is negligible.

3. Compare quantization with ignoring communication.

Table 2 shows that totally dropping communication will badly downgrade the model accuracy, though achieving slightly better speedup. PipeGCN (Wan et al., 2022b) and LLCG (Ramezani et al., 2022) also show that dropping all communication suffers from the slowest convergence and test score, which is not suggested in practice.

GraphSAGE			
Dataset (N=4)	Method	Throughput	Accuracy (%)
Reddit	0-bit	11.8	95.8
Reddit	1BG	8.6	97.1
Ogbn-products	0-bit	4.3	77.9
Ogbn-products	1BG	3.4	79.1

Table 2. Results of using 1BG and totally dropping communication.

4. Experiments only use GNNs with shallow layers.

As requested, we implement 1BG on APPNP with 10 propagation layers and evaluate on Reddit. The results are listed in Table 3. Our method still shows impressive speedup on deeper GNNs (11.3× on two servers) with negligible accuracy loss. When node number increases, the performance is further enhanced, which is consistent with the results in our paper. We will try to apply 1BG on other deeper and latest GNNs in future work.

Method	Epoch time (s)	Accuracy (%)
DGL (32-bit)	24.05 (1.0×)	44.89
1BG	2.13 (11.3×)	44.88

Table 3. Results of APPNP on Reddit on two servers.

2 To Reviewer 2

Thank you for your recognition of our work!

1. The argument for unbiasedness seems to miss the impact of the nonlinearity.

This is a valid concern, but analysis of the complete multi-layer GCN is difficult due to non-linear activations. When analyzing unbiasedness, we follow the same assumption as current works GraphSAINT (Zeng et al., 2020), FASTGCN (Chen et al., 2018), that each layer independently learns an embedding. Thus, the condition on the previous layer can be

removed. This assumption also motivates current edge sampler methods. For example, in Section 3.4 of GraphSAINT (Zeng et al., 2020), they have also performed analysis by assuming no non-linear activations. Then they can collapse L layers of $\boldsymbol{A}$ into an equivalent 1 layer of $\boldsymbol{A}^L$. Analysis based on this assumption leads to the proposed random walk sampler.

In practice, it is possible that neither of the above two assumptions are exactly true. Thus, we analyze the unbiased embedding under quantization of each layer independently. Our experiments in the paper also show that the quantization based on this assumption does lead to non-degraded accuracy.

2. Notation change in describing the elementwise nonlinearity.

As mentioned in Section-3.4, line 15 in Alg.1 is the matrix format of Equ.1 and 2, so the notation is different.

3. Choice of the nonlinearity sigma.

Sigma is not set to be a hyperparameter in our settings and we use Relu as the activation function.

4. There exist random quantization strategies with a certain property.

Yes, we assume the chosen strategy has this property.

5. What is the nonlinearity used in the experimental results?

The aggregation and update steps together form the nonlinear parts, and we use mean aggregator here. As for the nonlinear activation function, we choose Relu.

6. Word HALO.

We use this to represent boundary nodes as officially stated in DGL documents (`https://docs.dgl.ai/en/0.9.x/generated/dgl.distributed.load_partition.html`).

6. Typo and grammar issues.

Thanks for pointing them out! We have corrected them in our paper.

3 To Reviewer 3

1. The importance of 1-bit over 4-bit, and the utility of heavy compression schemes.

Sorry for our unclear statement of 1BG novelty. Specifically, 1BG differs from the referred 4-bit work in two aspects: (1) It quantizes **model weights** to accelerate model **inference**, while ours quantizes **communication** to accelerate model **training**. (2) GNN has distinct characteristics from traditional DNN models. Our key insight of this work is that GNN training has higher tolerance in reduced precision communication compared to DNNs, so extreme 1-bit is capable to maximize the training speed while maintaining accuracy (see Reviewer1-Q2). Fig.10 in our paper also shows lower bits can achieve higher throughput while the accuracy holds well on different bits.

Also, the overhead in GNN scope lies in communicating large amount of layer-wise data, so compression schemes are natural and effective to be applied. Optimized SGD methods such as Pipe-SGD (Li et al., 2018) common in DNNs cannot be grafted to our scenario since their overhead lies in all-reduce (green bar in Fig. 2).

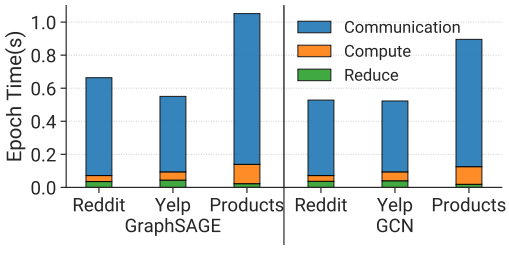

Figure 2. Training time per epoch and its breakdown using DGL.

2. Compare with MariusGNN.

Thanks for your constructive feedback! As requested, we run a simple experiment of training GraphSAGE on Ogbn-arxiv with these methods and results are in Table 4. Our method still outperforms in training speed since MariusGNN has large CPU-GPU data transfer overhead. MariusGNN achieves resource-efficient training and uses fewer GPUs, but with cost of

mixed CPU-GPU overhead. Our work is orthogonal to it and highlights the importance of GPU-oriented data communication so it can be incorporated into our work in the future.

Method	Throughput	Accuracy (%)
MariusGNN	12.8	69.1
1BG	18.2	69.0

Table 4. Results of MariusGNN and 1BG.

3. The theoretical part of the paper is also unclear.

Please refer to Reviewer1-Q1 and Q2.

4 To Reviewer 4

1. The idea of using quantization to reduce communication is well-studied.

We humbly disagree: First, we are the first to find GNN is more robust with reduced precision communication compared to DNNs, and utilize extreme 1-bit to reduce GPU-GPU communication while maintaining accuracy. For example, adopting 4-bit quantization to DNNs already causes significant accuracy drop, as shown in Table 5. Besides, current works in DNN apply quantization on the overhead of synchronizing all gradients and parameters (green bar in Fig. 2), while in GNN scope the overhead comes from layer-wise embeddings and feature gradients exchange (blue bar in Fig. 2), not all-reduce.

Method	ResNet18 (ImageNet)	ResNet50 (ImageNet)	GraphSAGE (Reddit)	GCN(Ogbn-products)
32-bit	71.2	77.1	97.1	73.1
4-bit	69.5	74.8	97.2	74.1

Table 5. Test accuracy (%) of applying quantization on DNN models and GNN models.

2. The theoretical analysis cannot guarantee the model accuracy could be well maintained.

Please refer to Reviewer1-Q2 part, we analyzed GNNs have better accuracy maintenance than DNNs so we can apply very low-bit precision on the communicated data without much accuracy loss. Our extensive experiments in the paper on multiple datasets and models also verify this.

3. Experiments are done with 1Gbps of Ethernet.

As requested, we conduct new evaluations on a cluster of four AWS EC2 g4dn.metal instances (96 vCPU, 384GB RAM, 8 T4 GPUs each) with fast 100Gbps network. We evaluate GraphSAGE on two datasets and list the training throughput in Table 6. Under such fast network setting, our method still shows impressive acceleration (2.0$\sim$3.3$\times$). Besides, it can exhibit and leverage the advantage of distributed training (more nodes higher speed) by reducing large communication overhead.

Method	Reddit (1 node)	Reddit (2 nodes)	Reddit (4 nodes)	Ogbn-products (1 node)	Ogbn-products (2 nodes)	Ogbn-products (4 nodes)
32-bit	1.95	1.96	1.77	1.3	1.79	1.87
1BG	4.68	5.41	5.73	2.61	4.27	5.13

Table 6. Throughput of 1BG on 100 Gbps network machines.

4. Partition method used.

Sorry for the omit. We adopt the classical partition method METIS in our work and all baselines are also based on METIS.

5. "Compute and reduce" time in Fig.1 in paper.

The higher compute and reduce time of some partitions come from all-reduce overhead caused by imbalance computation. Some partitions with smaller workloads have to wait for other partitions and the waiting time may dominate all-reduce overhead.

6. cuSPARSE lacks support for low-precision computations.

Sorry for the unprecise statement and we will correct it. Actually, 1BG currently supports FP16 computation but the speedup

of applying it is limited due to the small computation quantity (can be seen from Fig. 2, the orange bar only accounts for a small portion). So we leave the optimization of cuSPARSE integration as our future work.

7. Theoretical analysis of convergence with asynchronous pipeline.

As stated in section 3.3, asynchronous pipelining has also been utilized in PipeGCN(Wan et al., 2022b) and we reference the convergence analysis from theorem 3.1 in it.

(Convergence of GNN with pipeline) There exists a constant E such that for any arbitrarily small constant $\varepsilon > 0$, we can choose a learning rate $\eta = \frac{\sqrt{\varepsilon}}{E}$ and number of training iterations $T = \left(\mathcal{L}\left(\boldsymbol{W}_1\right) - \mathcal{L}\left(\boldsymbol{W}^*\right)\right) E\varepsilon^{-\frac{3}{2}}$ such that

$$\frac{1}{T}\sum_{t=1}^{T}\left\|\nabla\mathcal{L}\left(\boldsymbol{W}_t\right)\right\|_2 \leq \mathcal{O}(\varepsilon) \tag{1}$$

where $\mathcal{L}\left(\cdot\right)$ is the loss function, $\boldsymbol{W}_t$ and $\boldsymbol{W}^*$ represent the parameter at iteration t and the optimal parameter respectively. Therefore, the convergence rate of GNN training with asynchronous pipelining is $\mathcal{O}\left(T^{-\frac{2}{3}}\right)$. The detailed proof can be found in appendix A of PipeGCN (Wan et al., 2022b).

8. Test on larger datasets.

Thanks for your interest in further experiments. We adopt standard datasets/models for evaluation, which are also used in many works such as PipeGCN (Wan et al., 2022b), Dorylus (Thorpe et al., 2021) and BNS-GCN (Wan et al., 2022a). We will include larger and new datasets in our future work, and we are confident our method still performs well or even better on larger graphs.

9. Discussion about sampling-based methods.

Some sampling-based methods realize training with single GPU and break memory demand in a resource-efficient way, but they still suffer from accuracy loss as shown in Table 7 and giant CPU-GPU overhead (refer to the results of CPU-GPU method MariusGNN in Reviewer3-Q2). Therefore, we consider using full-graph training in a GPU-oriented way. But sampling-based methods are also orthogonal to our method so our method can certainly be incorporated into distributed sampling training in the future.

	Sampling-based			**Full-Graph**
Sample Size	5	10	15	
Accuracy (%)	73.55	74.87	76.84	**79.19**

Table 7. Test accuracy of sampling-based method and full-graph training.

10. Compare with CPU-GPU method Quiver.

As requested, we run GraphSAGE on Ogbn-products (N=4) and list results in Table 8. Our method achieves higher speedup and comparable accuracy than Quiver.

Method	Throughput	Accuracy (%)
Quiver	2.5	78.7
1BG	3.4	79.1

Table 8. Results of Quiver and 1BG.

REFERENCES

Lipschitz continuity. https://en.wikipedia.org/wiki/Lipschitz_continuity, 2022.

Bauer, F. L. Computational graphs and rounding error. *SIAM Journal on Numerical Analysis*, 11(1):87–96, 1974. doi: 10.1137/0711010. URL https://doi.org/10.1137/0711010.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *CoRR*, abs/1606.04838, 2018.

Chen, J., Ma, T., and Xiao, C. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, ICLR '18, 2018.

Li, Y., Yu, M., Li, S., Avestimehr, S., Kim, N. S., and Schwing, A. Pipe-sgd: A decentralized pipelined sgd framework for distributed deep net training. In *Advances in Neural Information Processing Systems*, NeurIPS '18, 2018.

Liu, Z., Zhou, K., Yang, F., Li, L., Chen, R., and Hu, X. Exact: Scalable graph neural networks training via extreme activation compression. In *International Conference on Learning Representations*, ICLR '21, 2021.

Ramezani, M., Cong, W., Mahdavi, M., Kandemir, M., and Sivasubramaniam, A. Learn locally, correct globally: A distributed algorithm for training graph neural networks. In *International Conference on Learning Representations*, ICLR '22, 2022.

Thorpe, J., Qiao, Y., Eyolfson, J., Teng, S., Hu, G., Jia, Z., Wei, J., Vora, K., Netravali, R., Kim, M., and Xu, G. H. Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads. In *15th USENIX Symposium on Operating Systems Design and Implementation*, OSDI '21, 2021.

Wan, C., Li, Y., Li, A., Kim, N. S., and Lin, Y. Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. In *Proceedings of Machine Learning and Systems*, MLSys '22, 2022a.

Wan, C., Li, Y., Wolfe, C. R., Kyrillidis, A., Kim, N. S., and Lin, Y. Pipegcn: Efficient full-graph training of graph convolutional networks with pipelined feature communication. In *International Conference on Learning Representations*, ICLR '22, 2022b.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, ICLR '20, 2020.