# KDD 2024 Rebuttal Reviewer wzvd

April 12, 2024

## 1 Reviewer wzvd

Dear reviewer, thank you so much for your valuable comments. We address all of your concerns as follows:

### 1.1 Q1

While doing forward pass F times will helps with feature staleness, the runtimes do not look promising. Methods such as Neighbor Sampling (GraphSAGE) or Layer-Neighbor Sampling [1] can train a 3-layer GraphSAGE model on ogbn-products in under a minute on a system with a decent GPU. They can train a 5-layer model in a slightly longer time, though still much faster than what is achieved in this work.

### 1.2 A1

It's important to note that running time depends on several factors such as batch size, the number of neighbors sampled per hop. To address your concern, we conducted experiments using 5-layer GraphSAGE models, consistent with our experiment settings in Table 5. We utilized the provided code scripts from the official repository to evaluate the total running time [1]. All experiments were conducted using a single GPU under identical conditions. To ensure a fair comparison, for SAGE, we tested two cases: (1) we used a batch size of 1024 following the setting in the official repository. (2) we used the same batch size as REST in our experiment, which is marked as "sage-large" in the table, but it resulted in out-of-memory (OOM) errors. Below, we present the efficiency analysis in Table 1 including running time until convergence.

From the results, it's worth noting that the running time of GraphSAGE is comparable to REST, however, it achieves lower performance due to the truncation of many neighbors. Additionally, the memory cost is much higher compared to GAS and REST.

The running time of GraphSAGE will increase significantly with the number of hops sampled, as neighbor sampling which is conducted in CPU plays a major role in determining the running time. In contrast, GAS and REST only sample one-hop neighbors, resulting in less sampling time. However, the running time

in historical embedding methods is primarily attributed to the movement of memory embeddings from the CPU to the GPU. Historical embeddings are stored in the CPU by default in GAS, and we follow the same approach in our implementation, so the running time can be further reduced by storing the memory embeddings directly in the GPU.

Importantly, we have to emphasis that the primary bottleneck in large-scale GNN training is achieving higher performance within the constraints of GPU memory limitations. We believe that relying solely on running time as a measure of model superiority is a biased evaluation criterion.

Table 1: Memory usage (MB) and running time (seconds) ogbn-products.

| **Dataset** | **Model** | TIME(S) | MEMORY(MB) | ACC(%) |
|---|---|---|---|---|
| ogbn-products | SAGE | 917 | 24412 | 78.8 |
| | SAGE-large | N/A | OOM | N/A |
| | REST | 940 | 13495 | 80 |
| | REST-IS | 1200 | 14753 | 80.4 |

[1] `https://github.com/pyg-team/pytorch_geometric/blob/master/examples/ogbn_products_sage.p`

### 1.3 Q2

Feature staleness can alternatively be reduced by reducing the learning rate by dividing it by F. Its effect is almost exactly the same as the proposed method. Thus, the proposed solution is an overengineered way of reducing the learning rate.

### 1.4 A2

We respectfully disagree with the comment. While there are some engineering techniques proposed to reduce feature staleness, such as adding regularization or reducing the learning rate, they often suffer from many potential issues during the training process. Taking reduced learning rate as an example:

(1) Reduced convergence speed and increased training time: Lowering the learning rate can slow down convergence and prolong training time significantly.

(2) Risk of getting stuck in local minima: Lower learning rates may cause the model to get trapped in local minima for extended periods, hindering overall optimization.

(3) Sensitivity to other hyperparameters: The effectiveness of reduced learning rates can depend heavily on other hyperparameter choices, such as the optimizer used.

In contrast, REST does not encounter the optimization issues associated with engineering techniques like reducing the learning rate. It involves only additional forward passes without changing the optimization process. Furthermore, REST is highly versatile: it can utilize higher frequencies or larger batch sizes to refresh the memory bank more frequently, thereby reducing staleness. Additionally, it can incorporate importance sampling to prioritize the refreshment of embeddings for important nodes, such as REST-IS. It's important to note that REST can achieve these outcomes that traditional engineering techniques cannot accomplish.

To further support our stance, we provide a comparison between REST with different frequencies (F) and simply reducing the learning rate by F. The original learning rate used in our experiment is 0.001. We provide performance, efficiency and convergence curve in the following table and figures to comprehensively support our claim. Upon examining the results, it's clear that when comparing F=2 with LR=0.0005 and F=5 with LR=0.0002, REST demonstrates superior performance and faster convergence rates compared to solely reducing the learning rate.

Table 2: Memory usage (MB) and running time (seconds) ogbn-products.

| **Dataset** | **Learning Rate** | Accuracy | Time(s) |
|---|---|---|---|
| ogbn-products | 0.0005 | 79.8 | 1950 |
| | 0.0002 | 79.7 | 2275 |
| **Dataset** | **Frequency** | Accuracy | Time(s) |
| ogbn-products | 2 | 80.1 | 1053 |
| | 5 | 80.4 | 1204 |

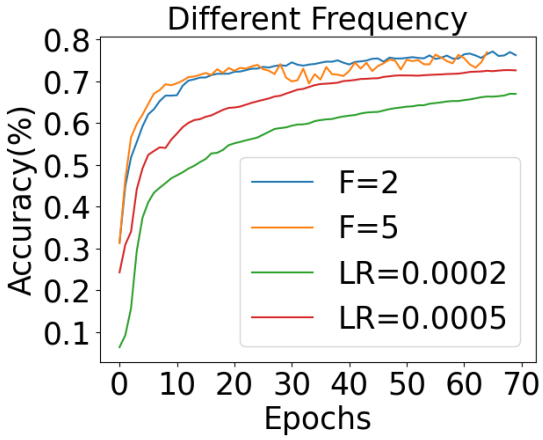


Figure 1: Convergence w.r.t epochs on the ogbn-products

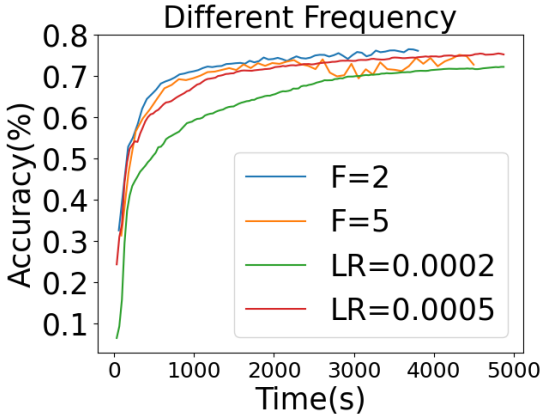


Figure 2: Convergence w.r.t time on ogbn-products

### 1.5 Q3

The used datasets are not the largest available, datasets such as papers100M, mag240M or the IGB [2] datasets could be used to be more convincing on the scalability aspect.

### 1.6 A3

We want to emphasize that our main baselines consist of historical embedding methods like VR-GCN, GAS, and GraphFM. Hence, we adhere to the experimental setups used in these methods, including the selection of datasets to ensure fair comparison. For small dataset, we choose to use smaller batch size to demonstrate the effectiveness of our proposed methods.

Following your recommendation, we carry out experiments using ogbn-papers100m, which is significantly larger in scale compared to other datasets. We report the performance and efficiency as follows.

Table 3: Memory usage (MB) and running time (seconds) ogbn-papers100m.

| **Models** | **Accuracy** | MEMORY(MB) | TIME(S) |
|---|---|---|---|
| GAS | 64.9 | 15705 | 8840 |
| REST | 67.3 | 16808 | 4100 |

The larger size of the ogbn-papers100M dataset exacerbates the staleness issue for GAS, leading to decreased accuracy and slower convergence, as anticipated. In contrast, REST showcases both high accuracy and efficiency, in line with our primary claim in the submission.