A Ablation Study

Here we show the ablation study over our method. Our proposed method contains two components: weighted kmeans to construct features for VR nodes $\bar{X}_{vr}$ using Eq 9 and computing propagation matrix $A_{vr,vr}$ using Eq 11. In Table 5, we show different variations of our method: VNG-kmeans is using normal kmeans to replace weighted kmeans; VNG-weighted-kmeans without using SVD and using least square instead to get $A_{vr,vr}$; then using different ranks when solving Eq 12 in VNG: VNG (rank=512), VNG (rank=128), VNG (rank=32) and VNG (rank=8). We learn from this table that weighted kmeans is performing better than kmeans (this verifies the usefulness of weighted kmeans to find $\bar{X}_{vr}$), and reducing the rank in VNG will decrease the performance but results in smaller graph size. So it is a trade-off when choosing the rank when approximating $A_{vr,vr}$ using low-rank method.

In Table 6, we conducted another ablation study that instead of solving a low-rank $A_{vr,vr}$, we use a sparse structure to reduce memory consumption. In this experiment, we compute the unconstrained optimal $A_{vr,vr}$ first and then prune part of its elements based on the absolute values. We observe that with 40% sparsity, the accuracy of VNG already reduces to 62.77% (compared with the original 67.14% accuracy). This suggests that low-rank compression works better than sparsity when we want to reduce the memory consumption of $A_{vr,vr}$.

Table 5: Ablation study on Arxiv dataset (in terms of accuracy).

VNG-kmeans	VNG-weighted-kmeans	VNG (rank=512)	VNG (rank=128)	VNG (rank=32)	VNG (rank=8)
66.57%(4.2MB)	67.15%(4.2MB)	67.14%(4.6MB)	67.07%(1.4MB)	66.48%(1.1MB)	66.25%(1.0MB)

B Proof of Theorem 1

Theorem 1 Assume $P = U_p \Sigma_p V_p^T$ and $Q = U_Q \Sigma_Q V_Q^T$ are the thin-SVD of$P$ and Q, respectively. Then the solution of

$$A_{vr,vr} = \arg\min_{H} \|HP - Q\|_F^2 \quad \text{s.t.} \quad \text{rank}(H) \leq k, \tag{13}$$

is $(QV_p)_k \Sigma_p^{-1} U_p^T$, where $(\cdot)_k$ denotes the rank-k truncated SVD of the matrix.

Proof:

$$\begin{aligned} A_{vr,vr} &= \arg\min_{H:\text{rank}(H)\leq k} \|HP - Q\|_F^2 \\ &= \arg\min_{H:\text{rank}(H)\leq k} \|(HP - Q)V_p V_p^T\|_F^2 + \|(HP - Q)(I - V_p V_p^T)\|_F^2 \\ &= \arg\min_{H:\text{rank}(H)\leq k} \|(HP - Q)V_p V_p^T\|_F^2 \\ &= \arg\min_{H:\text{rank}(H)\leq k} \|(HP - Q)V_p\|_F^2 \\ &= \arg\min_{H:\text{rank}(H)\leq k} \|HU_p\Sigma_p - QV_p\|_F^2 \\ &= (QV_p)_k \Sigma_p^{-1} U_p^T. \end{aligned} \tag{14}$$

C Preprocessing time for our method

Our method mainly consists of three steps: (1) Inference of the training data to obtain their embeddings; (2) performing weighted k-means (Section 3.1); and (3) performing low-rank approximation (Section 3.2).

For step (1) we only need to perform the forward pass of the training data once to obtain the embeddings for each layer. For our largest Amazon2M dataset, it takes 11.1 seconds.

For step (2), weighted k-means takes $O(ncdT)$ time for clustering, where T is the number of steps for weighted-k-means, typically <20; c is the number of clusters (virtual nodes); d is feature dimension

Table 6: Ablation study on Arxiv dataset (in terms of accuracy) using sparse vs low-rank for $A_{vr,vr}$.

VNG (rank=512)	VNG (rank=8)	VNG-Sparse (sp: 90%)	VNG-Sparse (sp: 40%)	VNG-Sparse (sp: 10%)
67.14%(4.6MB)	66.25%(1.0MB)	58.89%(1.4MB)	62.70%(4.4MB)	66.72%(6.4MB)

and n is number of training nodes. When c is large, we can use hierarchical k-means so the complexity will be roughly $O(ndT * \log(c))$. For Amazon2M dataset with 0.5% compression rate, it takes 3234.8 seconds. This step could also be sped up by sampling of training data.

For step (3), we compute a closed form solution for (12) in Theorem 1. Here we analyze the time complexity and show the run time for achieving such low-rank closed form solution. The main computation in Theorem 1 is the SVD of two tall and thin matrices P and Q which are c-by-d matrices (where c is the number of compressed nodes and d is the hidden dimensions times the number of GNN layers plus input dimension). In practice, the size of c depends on the desired compression rate, and d is fixed based on the GNN architecture. The time complexity for computing the SVD of a c-by-d matrix is $O(\min(c,d)^2 \max(c,d))$, which will be fast if one of the c, d is small. For example, when $d < c$, the time complexity for computing SVD of P (or Q) will be $O(d^2c + d^3)$ which is linear to c. This is using the fact that since $P = USV^T$, then $P^TP = VS^2V^T$, so they share the same right singular vectors. Therefore, we can compute SVD of P^TP (d-by-d matrix) instead of P. More specifically, the procedure will be 1) Compute $H = P^TP$ which takes $O(d^2c)$ time, 2) Compute SVD of $H = V\Sigma V^T$ which takes $O(d^3)$ time. 3) Compute $S = \Sigma^{1/2}$ which is a diagonal matrix and this costs $O(d)$ time; 4) Compute $U = PVS^{-1}$ which takes $O(d^2c)$ time.

Typically in GNN d is small. For example, $d = 2648$ for Reddit. If we use 5% compression rate on Reddit, then $c = 7696$. In this case, running the computation we mentioned above for SVD computation will take only 4 seconds on a CPU, even without using a GPU. For our largest Amazon2M dataset with 0.5% compression rate, this step takes 44.4 seconds.

D VIRTUALIZATION OF VIRTUAL NODES

In Figure 3, we show the 2-D projection of the virtual nodes (red dots) and training nodes (blue dots) using PCA over graph node features. This figure is based on the data in the Reddit dataset, where we generate 10 virtual nodes with our algorithm VNG, and randomly select 1000 training nodes for visualization purpose.

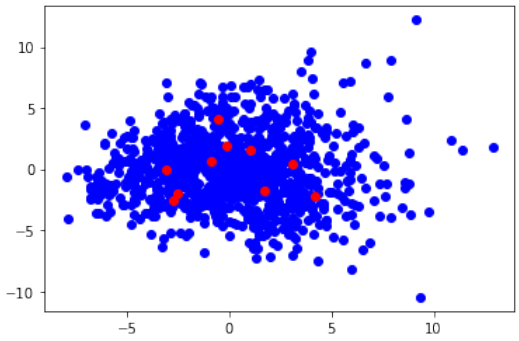

Figure 3: 2-D projection of the virtual nodes (red dots) and training nodes (blue dots) using PCA over their node features on the Reddit dataset.

E EXPERIMENTAL RESULT ON GRAPHSAGE MODEL

Besides GCN model, our method can also be applied to other GNN models to reduce inference model serving size. In Table 7, we show the performance of various graph compression methods on GraphSAGE model (Hamilton et al., 2017) with mean aggregator over Arxiv dataset. The results show that our method is still consistently better than the baselines.

Table 7: Results on GraphSAGE model on Arxiv dataset. We construct the compressed graph with only 1% of total training nodes for our method. We use the graph size of our proposed method (VNG) as the baseline (1x), and other methods' graph sizes are evaluated w.r.t. this baseline. For example, 10x means using 10 times more space as compared to our method. 'Full' means using the original adjacency graph and nodes features.

Full	SVD	Sparse	Random	Degree	Kmeans	VNG
69.59(37.5x)	59.1(5.8x)	56.3(4.5x)	57.2(1.0x)	62.6(1.3x)	57.6(1.0x)	65.9(1.0x)

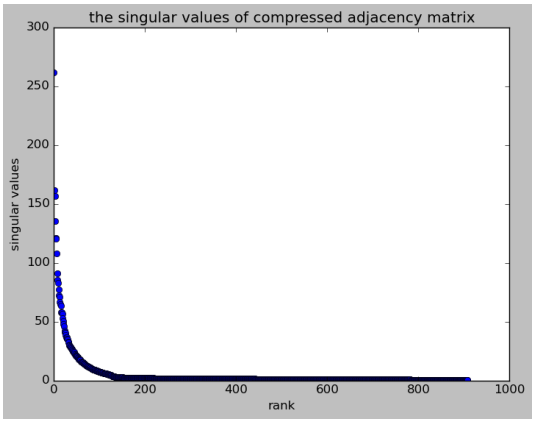

Figure 4: The singular values plot of learned $A_{vr,vr}$. We can see when the rank increases, the singular values will decrease dramatically, which shows that some low-rank structure exists in $A_{vr,vr}$.

F SINGULAR VALUES OF $A_{vr,vr}$

In Figure 4, we show the singular values of $A_{vr,vr}$, where $A_{vr,vr}$ is the unconstrained optimal. We compute $A_{vr,vr}$ on Arxiv dataset with 1% VR nodes. We observe that the $A_{vr,vr}$ matrix has a low-rank structure, validating why our low-rank approach works well in practice.

G INFERENCE SPEED

Although we focus on reducing the space complexity in this paper, our method will not sacrifice inference speed. Instead of storing full $A_{vr,vr}$, we store it in the factorization form where $A_{vr,vr} = UV^T$ and U, V are obtained by Theorem 1. Let k be the rank, c be the number of VR nodes and d be the feature dimension, our method $A_{vr,vr}X_{vr} = U(V^T X_{vr})$ takes $O(cdk)$ flops while the original propagation $A_{tr,tr}X_{tr}$ takes $O(\text{nnz}(A)d)$ flops. In most of the experiments in our paper, we achieve a better or at least comparable inference speed. To compare the inference speed, we run it on a standard CPU with full-batch inference on the whole testing set, and report the numbers in Table 8.

Table 8: Inference time (in seconds). Since ClusterGCN's inference is on CPU, the below numbers are based on CPU time. *For Amazon2M, #nodes in the compressed graph is 0.1% or 0.5% of original graph.

Datasets		Arxiv	Reddit	Product	Amazon2M*
Full		1.28	20.6	8.351	65.24
VNG	1%	0.57	1.64	2.95	13.88
	5%	1.17	12.08	7.64	16.38

Table 9: Accuracy% (Size) on Arxiv, Reddit and Product datasets for single node setting where testing data appears one-by-one setting. 'Full' corresponds to using the original adjacency graph and nodes features. In this table, we report the compression rate of VNG until its accuracy is within 1% of the full model, and show the accuracy of other methods under the same compression rate.

Model	Node Compression rate	Full	VNG	Random	Degree	Kmeans
Arxiv	5%	68.35	67.84	58.57	63.95	57.75
Reddit	15%	92.75	91.98	81.02	87.04	83.07
Product	10%	86.95	86.18	76.37	80.09	74.73