CONSTANT TIME GRAPH NEURAL NETWORKS

Ryoma Sato
Kyoto University
r.sato@ml.ist.i.kyoto-u.ac.jp

Makoto Yamada
Kyoto University, RIKEN AIP, JST PRESTO
myamada@i.kyoto-u.ac.jp

Hisashi Kashima
Kyoto University, RIKEN AIP
kashima@i.kyoto-u.ac.jp

ABSTRACT

Recent advancements in graph neural networks (GNNs) have led to state-of-the-art performance in various applications including chemo-informatics, question answering systems, and recommendation systems, to name a few. However, making these methods scalable to huge graphs is challenging. In particular, the existing methods for accelerating GNNs are either not theoretically guaranteed in terms of approximation error or require at least linear time computation cost. In this paper, we propose a constant time approximation algorithm for the inference and training of GNNs that theoretically guarantees arbitrary precision with arbitrary probability. The key advantage of the proposed algorithm is that the complexity is completely independent of the number of nodes, edges, and neighbors of the input. To the best of our knowledge, this is the first constant time approximation algorithm for GNNs with theoretical guarantee. Through experiments using synthetic and real-world datasets, we evaluate our proposed approximation algorithm and show that the algorithm can successfully approximate GNNs in constant time.

1 INTRODUCTION

Machine learning on graph structures has many applications such as chemo-informatics (Gilmer et al., 2017), drug discovery (Li et al., 2018; You et al., 2018), question answering systems (Nickel et al., 2016; Schlichtkrull et al., 2017; Hamilton et al., 2018), recommendation systems (Ying et al., 2018), and physical systems (Sanchez-Gonzalez et al., 2018) to name a few. Recently, a novel graph learning algorithm called graph neural networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009; Kipf & Welling, 2016; Hamilton et al., 2017) was proposed, and it showed state-of-the-art performances in various graph learning tasks. However, GNNs need at least $\deg(v)$ operations to aggregate neighbor features to the node v. Since many real-wold data follow the power-law and some hub nodes can have extremely high degrees (*e.g.,* celebrities in social networks), it is computationally expensive to aggregate neighbor features of such high degree nodes. Moreover, if we consider 2 or more hops (*i.e.,* layers), even an ordinary node has to aggreagte a large number of features since it connects with some hub nodes. Therefore, applying GNNs to huge graphs is challenging. Although Ying et al. (2018) succeeded in applying GNNs to a web-scale network by using MapReduce, it still requires massive computational resources.

There are several node sampling techniques for reducing GNN computation. For example, an empirical neighbor sampling scheme is used to speed up GraphSAGE (Hamilton et al., 2017). FastGCN (Chen et al., 2018b), Huang et al. (2018) and Chen et al. (2018a) proposed other sampling methods to accelerate GNNs. Overall, the existing sampling techniques for GNNs work well in practice. However, these techniques are either not theoretically guaranteed in terms of approximation error or require at least linear time computation cost for training and inference GNNs.

In this paper, we utilize neighbor sampling (Hamilton et al., 2017) and derive a constant time approximation algorithm for inference and gradient computation of GNNs. To be precise, given an error tolerance ε and confidence probability $1-\delta$, our approximation algorithm computes the estimate $\hat{\boldsymbol{z}}_v$ of the exact embedding $\boldsymbol{z}_v$ of a node v, such that $\Pr[\|\hat{\boldsymbol{z}}_v - \boldsymbol{z}_v\|_2 \geq \varepsilon] \leq \delta$ and the estimate $\widehat{\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}}}$

Table 1: ✓ means that the network can be approximated in constant time *by our proposed algorithm*. ✗ means that the network cannot be approximated *by any algorithm.*

	GraphSAGE-GCN		GraphSAGE-mean		GraphSAGE-pool	GCN
Activation	Inference	Training	Inference	Training		
sigmoid / tanh	✓ Thm. 1	✓ Thm. 3	✓ Thm. 1	✓ Thm. 3	✗ Thm. 8	✗ Thm. 9
ReLU	✓ Thm. 1	✗ Thm. 7	✓ Thm. 1	✗ Thm. 7	✗ Thm. 8	✗ Thm. 9
ReLU + normalization	✗ Thm. 6	✗ Thm. 6	✗ Thm. 6	✗ Thm. 6	✗ Thm. 8	✗ Thm. 9

Algorithm 1 $\mathcal{O}_z$: Exact embedding

Require: Graph $G = (V, E)$ (as oracle); Feature $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$ (as oracle); Weight matrix $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}} (l = 1, \dots, L)$; Node index $i \in V$.
Ensure: Exact embedding $\boldsymbol{z}_i$

$\boldsymbol{z}_v^{(0)} \leftarrow \boldsymbol{x}_v \ (\forall v \in V)$ # initialize embedding as feature vectors
for $l = 1, \dots, L$ **do**
  $\boldsymbol{m}_v^{(l)} \leftarrow \frac{1}{deg(v)} \sum_{u \in \mathcal{N}(v)} \boldsymbol{z}_u^{(l-1)} \ (\forall v \in V)$ # aggregate features
  $\boldsymbol{h}_v^{(l)} \leftarrow \boldsymbol{W}^{(l)} \boldsymbol{m}_v^{(l)} \ (\forall v \in V)$ # linear transformation
  $\boldsymbol{z}_v^{(l)} \leftarrow \sigma(\boldsymbol{h}_v^{(l)}) \ (\forall v \in V)$ # apply activation function
end for
return $\boldsymbol{z}_i^{(L)}$

of the exact gradient $\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}}$ of the embedding $\boldsymbol{z}_v$ with respect to the network parameters $\boldsymbol{W}^{(l)}$, such that $\Pr[\|\widehat{\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}}\|_F \geq \varepsilon] \leq \delta$. Our algorithm can approximate the exact embedding and its gradients within $O(\frac{1}{\varepsilon^{2L}}(\log \frac{1}{\varepsilon} + \log \frac{1}{\delta})^{L-1}(\log \frac{1}{\delta}))$ time, where L is the number of layers. This complexity is completely independent of the number of nodes, edges, and neighbors of the input; the proposed algorithm can deal with graphs irrespective of however large they may be.

2 Background

Notations: Let G be the input graph, $V = \{1, 2, \dots, n\}$ be the set of nodes, $n = |V|$ be the number of nodes, E be the set of edges, $m = |E|$ be the number of edges, $\deg(v)$ be the degree of a node v, $\mathcal{N}(v)$ be the set of neighbors of a node v, $\boldsymbol{x}_v \in \mathbb{R}^{d_0}$ be the feature vector associated to a node $v \in V$, $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2 \dots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times d_0}$ be the stacked feature vectors, and $^\top$ denotes the matrix transpose. We assume there is always a self loop for each node.

Node Embedding Model: We consider graph embedding problems and employ GraphSAGE-GCN (Hamilton et al., 2017) to calculate the embeddings of nodes. The same arguments about approximability are applicable to GraphSAGE-mean with a slight modification. Algorithm 1 shows the pseudo code, where $\sigma(\cdot)$ is an elementwise activation function (*e.g.,* sigmoid, ReLU). The final output is simply denoted as $\boldsymbol{z}_i = \boldsymbol{z}_i^{(L)}$. Note that, we do not normalize $\boldsymbol{z}^{(l)}$ as in the original network because GraphSAGE-GCN cannot be approximated in constant time with normalization (see Theorem 6).

Computational Model Assumptions: We have to specify how to access the input to design constant time algorithms because the constant time algorithms cannot read the entire input. In this paper, we follow the standard convention of sublinear time algorithms (Parnas & Ron, 2007; Nguyen & Onak, 2008). Namely, we model our algorithm as an oracle machine that can query about the input and measure the complexity by query complexity. Algorithms can access the input only by querying the following three oracles: (1) $\mathcal{O}_{\text{deg}}(v)$: the degree of node v, (2) $\mathcal{O}_G(v, i)$: the i-th neighborhood of node v, and (3) $\mathcal{O}_{\text{feature}}(v, i)$: the i-th feature of node v.

Algorithm 2 $\hat{\mathcal{O}}_z^{(1)}$: Constant time approximation of $\boldsymbol{z}_v^{(1)}$

Require: Graph $G = (V, E)$ (as oracle); Feature $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$ (as oracle); Weight matrix $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}} (l = 1, \ldots, L)$; Node index $v \in V$; Error tolerance ε; Confidence probability $1 - \delta$.

Ensure: Approximation of the embedding $\boldsymbol{z}_v^{(1)}$

1: $r^{(1)} \leftarrow \mathrm{ceil}(\frac{2B^2 B_{\mathrm{op}}^2 K^2 d_0}{\varepsilon^2} \log \frac{2d_0}{\delta})$
2: $S^{(1)} \leftarrow$ uniformly sample with replacement from the neighbors of v with $r^{(1)}$ samples
3: $\hat{\boldsymbol{m}}_v^{(1)} \leftarrow \frac{1}{r^{(1)}} \sum_{u \in S^{(1)}} \boldsymbol{x}_u$
4: $\hat{\boldsymbol{h}}_v^{(1)} \leftarrow \boldsymbol{W}^{(1)} \hat{\boldsymbol{m}}_v^{(1)}$
5: $\hat{\boldsymbol{z}}_v^{(1)} \leftarrow \sigma(\hat{\boldsymbol{h}}_v^{(1)})$
6: **return** $\hat{\boldsymbol{z}}_i$

Algorithm 3 $\hat{\mathcal{O}}_z^{(l)}$: Constant time approximation of $\boldsymbol{z}_v^{(l)}$

Require: Graph $G = (V, E)$ (as oracle); Feature $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$ (as oracle); Weight matrix $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}} (l = 1, \ldots, L)$; Node index $v \in V$; Error tolerance ε; Confidence probability $1 - \delta$.

Ensure: Approximation of the embedding $\boldsymbol{z}_v^{(l)}$

1: $r^{(l)} \leftarrow \mathrm{ceil}(\frac{8B_{l-1}^2 B_{\mathrm{op}}^2 K^2 d_{l-1}}{\varepsilon^2} \log \frac{4d_{l-1}}{\delta})$ # $B_0 = B, B_l = K B_{\mathrm{op}} B_{l-1} + \|\sigma(\boldsymbol{0}_{d_l})\|_2$
2: $S^{(l)} \leftarrow$ uniformly sample with replacement from the neighbors of v with $r^{(l)}$ samples
3: $\hat{\boldsymbol{m}}_v^{(l)} \leftarrow \frac{1}{r^{(l)}} \sum_{u \in S^{(l)}} \hat{\mathcal{O}}_z^{(l-1)}(u, \frac{\varepsilon}{4KB_{\mathrm{op}}}, \frac{\varepsilon\delta}{16B_{\mathrm{op}}KB_{l-1}})$
4: $\hat{\boldsymbol{h}}_v^{(l)} \leftarrow \boldsymbol{W}^{(l)} \hat{\boldsymbol{m}}_v^{(l)}$
5: $\hat{\boldsymbol{z}}_v^{(l)} \leftarrow \sigma(\hat{\boldsymbol{h}}_v^{(l)})$
6: **return** $\hat{\boldsymbol{z}}_i$

Problem Formulation: Under the fixed network structure (*i.e.,* the number of layers L, the dimensions d_l, and the activation function $\sigma(\cdot)$), we construct theoretically guaranteed approximation algorithms that calculate the approximation of embedding vectors $\boldsymbol{z}_v$ and gradients $\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}}$ in constant time irrespective of the number of nodes, edges, neighbors of the input and the feature vectors $\boldsymbol{X}$ and network parameters $\boldsymbol{W}^{(l)}$. However, it is impossible to construct a constant time algorithm without any assumption about the inputs. Therefore, we make some mild assumptions:

Assumption 1 $\|\boldsymbol{x}_i\|_2$ and $\|\boldsymbol{W}^{(l)}\|_{\mathrm{op}}$ are bounded by some constants B and B_{op}, respectively.

Assumption 2 The activation function $\sigma(\cdot)$ is K-Lipschitz (e.g., sigmoid, ReLU).

Assumption 3 The gradient of the activation function $\sigma'(\cdot)$ is K'-Lipschitz (e.g., sigmoid, tanh).

We use Assumption 3 only for gradient computation. Note that if $\|\boldsymbol{W}^{(l)}\|_{\mathrm{op}}$, the operator norm of $\boldsymbol{W}^{(l)}$, is bounded by some constant B_{op}, then $\|\boldsymbol{W}^{(l)}\|_F$ is also bounded by some constant B_F. We prove later that it is impossible to construct a constant time algorithm without these assumptions (see Section 4 for details). Assumption 1 can be achieved by feature clipping, weight clipping, or weight decay in practice.

3 Proposed Method

Constant time embedding approximation (Inference): Here, we propose a constant time approximation algorithm based on neighbor sampling, which approximates the embedding $\boldsymbol{z}_v$ with an absolute error of at most ε and probability $1 - \delta$. We construct the algorithm layer by layer recursively. We denote the algorithm that calculates the estimate of embeddings in the l-th layer $\boldsymbol{z}^{(l)}$ as $\hat{\mathcal{O}}_z^{(l)}(l = 1, \ldots, L)$. We show the pseudo codes of the base case in Algorithm 2 and the inductive step in Algorithm 3. In the following, we prove that Algorithms 2 and 3 approximate the embedding $\boldsymbol{z}_v$ with arbitrary precision and arbitrary probability.

Theorem 1. *Under Assumptions 1 and 2,* $\forall \varepsilon > 0, 0 < \delta < 1$,

$$\Pr[\|\hat{\mathcal{O}}_z^{(L)}(v, \varepsilon, \delta) - \boldsymbol{z}_v\|_2 \geq \varepsilon] \leq \delta.$$

Theorem 1 shows that Algorithms 2 and 3 approximate the embedding $\boldsymbol{z}_v$ with arbitrary precision and arbitrary probability.

Next, we will investigate the query complexity of Algorithms 2 and 3.

Theorem 2. *Under Assumptions 1 and 2, the query complexity of Algorithms 2 and 3 is* $O(\frac{1}{\varepsilon^{2L}}(\log\frac{1}{\varepsilon}+\log\frac{1}{\delta})^{L-1}(\log\frac{1}{\delta}))$.

Theorem 2 shows that we can approximate the exact embedding within $O(\frac{1}{\varepsilon^{2L}}(\log\frac{1}{\varepsilon}+\log\frac{1}{\delta})^{L-1}(\log\frac{1}{\delta}))$ time, and this complexity is independent of the the number of nodes, edges, and neighbors of the input; the proposed algorithm works completely irrespective of how large the input graphs are. Moreover, The complexity is polynomial with respect to $\frac{1}{\varepsilon}$ and $\log\frac{1}{\delta}$.

Constant time gradient approximation (Training): Next, we propose a constant time approximation algorithm that approximates the gradient of embeddings with respect to the network parameters with an absolute error of at most ε and probability $1-\delta$. The basic strategy is to run Algorithms 2 and 3, and then calculate the gradients of the embedding $\boldsymbol{z}_v$. Let $\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}}$ be the gradient of the embedding $\boldsymbol{z}_v$ with respect to the network parameter $\boldsymbol{W}^{(l)}$ (*i.e.,* $(\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}^{(l)}})_{ijk}=\frac{\partial \boldsymbol{z}_{vi}}{\partial \boldsymbol{W}^{(l)}_{jk}}$).

Theorem 3. *Let* $\widehat{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}}(\varepsilon,\delta)$ *be the gradient of* $\hat{\boldsymbol{z}}_v^{(L)}$*, which is obtained by running* $\hat{\mathcal{O}}_z^{(L)}(v,\varepsilon,\delta)$ *and backpropagation, with respect to* $\boldsymbol{W}^{(l)}$*. Then, under Assumptions 1, 2, and 3, there exist constants* C *and* D *such that* $\forall\varepsilon>0, 0<\delta<1$*,*

$$\Pr[\|\widehat{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}}(C\varepsilon,D\delta)-\frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F\geq\varepsilon]\leq\delta.$$

Corollary 4. *Under Assumptions 1, 2, and 3, we can calculate the gradient* $\frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}$ *with an absolute error of at most* ε *and probability* $1-\delta$ *in* $O(\frac{1}{\varepsilon^{2L}}(\log\frac{1}{\varepsilon}+\log\frac{1}{\delta})^{L-1}(\log\frac{1}{\delta}))$ *time.*

4 INAPPROXIMABILITY

In this section, we show that some existing GNNs cannot be approximated in constant time by our proposed algorithm but also *by any other algorithm*. In other words, for any algorithm that has the error tolerance ε and confidence probability δ as parameters and calculates an estimate of embedding or gradient of some classes of GNNs in constant time, there exist $\varepsilon>0, \delta>0$ and an input such that satisfies assumptions but the approximation error is at least ε with probability at least δ.

Theorem 5. *Without Assumption 1, the inference of GraphSAGE-GCN (Hamilton et al., 2017) with any activation but a constant function cannot be approximated with arbitrary precision and probability in constant time.*

Theorem 6. *Even under Assumption 1, the inference and gradients of GraphSAGE-GCN (Hamilton et al., 2017) with ReLU activation and normalization cannot be approximated with arbitrary precision and probability in constant time.*

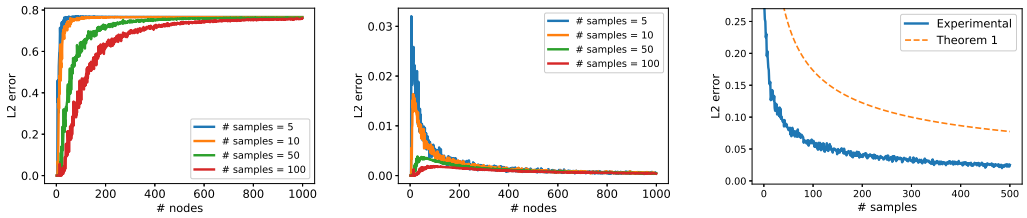
Theorem 7. *Even under Assumptions 1 and 2, the gradients of GraphSAGE-GCN (Hamilton et al., 2017) with ReLU activation cannot be approximated with arbitrary precision and probability in constant time.*

Note that the *inference* of GraphSAGE-GCN with ReLU activation (without normalization layer) *can* be approximated in constant time by using our algorithm (Theorem 1).

The following two theorems state that these networks cannot be approximated in constant time even under Assumptions 1, 2, and 3.

Theorem 8. *Even under assumptions 1, 2, and 3, the inference of GraphSAGE-pool (Hamilton et al., 2017) cannot be approximated with arbitrary precision and probability in constant time.*

Theorem 9. *Even under Assumptions 1, 2, and 3, the inference of GCN (Kipf & Welling, 2016) cannot be approximated with arbitrary precision and probability in constant time.*



(a) Original. (b) Sigmoid. (c) Bound.

Figure 1: Synthetic data experiments. (a) The approximation error of the original 1-layer GraphSAGE-GCN (*i.e.,* with ReLU activation and normalization) with synthetic data. (b) The approximation error of 1-layer GraphSAGE-GCN with sigmoid activation using the same synthetic data as that in (a). (c) The approximation error of 1-layer GraphSAGE-GCN with sigmoid activation using synthetic data and its theoretical bound.

5 Experiments

In this section, we confirm the following two facts through computational experiments:

Fact 1: The embeddings can be well approximated in constant time by Algorithms 2 and 3 (Theorem 1), whereas the original GraphSAGE-GCN cannot be approximated in constant time (Theorem 6).

Fact 2: Algorithm 2 is efficient.

We confirm Facts 1 and 2 with synthetic data. First, we use the original 1-layer GraphSAGE-GCN (with ReLU activation and normalization) and 1-layer GraphSAGE-GCN with sigmoid activation for comparison. The input graph is a clique K_n, the features are $\boldsymbol{x}_1 = [1, 0]^\top$ and $\boldsymbol{x}_i = [0, 1/n]^\top$ $(i \neq 1)$, and the weight matrix is an identity matrix $\boldsymbol{W}^{(1)} = \boldsymbol{I}_2$. We use $r^{(1)} = 5, 10, 50, 100$ as the sample size. If a network can be approximated in constant time, the approximation error goes to zero as the sample size increases even if the graph size goes to infinity. We show the approximation errors of both the networks in Figure 1 (a) (b). The approximation error of the original GraphSAGE-GCN converges to about 0.75 even if the sample size increases. On the other hand, the approximation error of GraphSAGE-GCN with sigmoid function goes to zero and the errors become increasingly bounded as the sample size increases. It matches Theorems 1 and 6.

Next, we use 1-layer GraphSAGE-GCN with sigmoid activation. The input graph is a clique K_n, where the number of nodes is $n = 1000$. We set the dimensions as $d_0 = d_1 = 2$ and each feature value is set to 1 with probability 0.5 and -1 otherwise. We initialize the weight matrix $\boldsymbol{W}^{(1)}$ with normal distribution and then normalize it so that $\|\boldsymbol{W}^{(1)}\|_{\mathrm{op}} = 1$. For each $r = 1 \ldots 500$, we (1) initialize the weight matrix; (2) calculate the exact embedding of each node; (3) calculate the approximation embedding of each node with r samples (*i.e.,* $r^{(1)} = r$); and (4) calculate the approximation error of each node. We show the 99-th percentile point and the theoretical bound by Theorem 1 with $\delta = 0.01$ in Figure 1 (c). It matches Theorems 1. Note Theorem 1 holds for any input, therefore the theoretical curve has to be above the experimental curve whatever the input is.

6 Conclusion

We proposed a constant time approximation algorithm for the inference and gradient computation of GNNs, where the complexity is completely independent of the number of nodes, edges, and neighbors of the input. We proved its theoretical guarantee in terms of the approximation error. This is the first constant time approximation algorithm for GNNs in the literature. We further showed that some existing GNNs cannot be approximated in constant time by any algorithm.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 15H01704. MY is supported by the JST PRESTO program JPMJPR165A.

REFERENCES

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 941–949, 2018a.

Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the Sixth International Conference on Learning Representations, ICLR 2018*, 2018b.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pp. 1263–1272, 2017.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2005*, volume 2, pp. 729–734, 2005.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 1025–1035, 2017.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 2030–2041, 2018.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, 2018.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *CoRR*, abs/1803.03324, 2018.

Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pp. 327–336, 2008.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 1955–1961, 2016.

Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin A. Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 4467–4476, 2018.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103, 2017.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, pp. 974–983, 2018.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 5694–5703, 2018.

A PROOFS

Lemma 10. *Let $B_0 = B$ and $B_l = KB_{op}B_{l-1} + \|\sigma(\mathbf{0}_{d_l})\|_2$ $(l = 1, \ldots, L)$, where $\mathbf{0}_{d_l}$ is the d_l dimensional zero vector. Under Assumptions 1 and 2, the norm of the embedding in each layer $\|\boldsymbol{z}_v^{(l)}\|_2 (l = 1, \ldots, L)$ is bounded by B_l.*

Proof of Lemma 10. $\|\boldsymbol{z}_v^{(l)}\|_2 = \|\sigma(\boldsymbol{h}_v^{(l)})\|_2 \leq \|\sigma(\boldsymbol{h}_v^{(l)}) - \sigma(\mathbf{0}_{d_l})\|_2 + \|\sigma(\mathbf{0}_{d_l})\|_2 \leq K\|\boldsymbol{h}_v^{(l)} - \mathbf{0}_{d_l}\|_2 + \|\sigma(\mathbf{0}_{d_l})\|_2 = K\|\boldsymbol{W}^{(l)}\boldsymbol{m}_v^{(l)}\|_2 + \|\sigma(\mathbf{0}_{d_l})\|_2 \leq KB_{\text{op}}\|\boldsymbol{m}_v^{(l)}\|_2 + \|\sigma(\mathbf{0}_{d_l})\|_2 \leq KB_{\text{op}}B_{l-1} + \|\sigma(\mathbf{0}_{d_l})\|_2 = B_l$. Therefore, $\|\boldsymbol{z}_v^{(l)}\|_2 \leq B_l$ holds inductively. □

Lemma 11 (Hoeffding (1963))**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables bounded by the intervals $[-B, B]$, and let $\bar{X}$ be the empirical mean of these variables $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$. Then, for any $\varepsilon > 0$,*

$$\Pr[|\bar{X} - \mathbb{E}[\bar{X}]| \geq \varepsilon] \leq 2\exp\left(-\frac{n\varepsilon^2}{2B^2}\right)$$

holds.

Lemma 12 (multivariate Hoeffding's inequality)**.** *Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ be independent d dimensional random variables whose 2-norms are bounded $\|\boldsymbol{x}_i\|_2 \leq B$, and let $\bar{\boldsymbol{x}}$ be the empirical mean of these variables $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$. Then, for any $\varepsilon > 0$,*

$$\Pr[\|\bar{\boldsymbol{x}} - \mathbb{E}[\bar{\boldsymbol{x}}]\|_2 \geq \varepsilon] \leq 2d\exp\left(-\frac{n\varepsilon^2}{2B^2 d}\right)$$

holds.

Proof of Lemma 12. Apply Lemma 11 to each dimension k of X_i. Then

$$\Pr[|\bar{X}_k - \mathbb{E}[\bar{X}]_k| \geq \frac{\varepsilon}{\sqrt{d}}] \leq 2\exp\left(-\frac{n\varepsilon^2}{2B^2 d}\right)$$

Note that $|X_{ik}| < B$ because $\|X_i\|_2 < B$. Therefore,

$$\Pr[\exists k \in \{1, 2, \ldots, d\}\ |\bar{X}_k - \mathbb{E}[\bar{X}]_k| \geq \frac{\varepsilon}{\sqrt{d}}] \leq 2d\exp\left(-\frac{n\varepsilon^2}{2B^2 d}\right)$$

If $|\bar{X}_k - \mathbb{E}[\bar{X}]_k| < \frac{\varepsilon}{\sqrt{d}}$ holds for all dimension k, then

$$\|\bar{X} - \mathbb{E}[\bar{X}]\|_2 = \sqrt{\sum_{k=1}^d (\bar{X}_k - \mathbb{E}[\bar{X}]_k)^2} < \sqrt{d \cdot \frac{\varepsilon^2}{d}} = \varepsilon$$

Therefore,

$$\Pr[\|\bar{X} - \mathbb{E}[\bar{X}]\|_2 \geq \varepsilon] \leq 2d\exp\left(-\frac{n\varepsilon^2}{2B^2 d}\right)$$

□

Lemma 13. *Let $A_1, A_2, \ldots A_n$ be probabilistic events such that $\Pr[A_i] \leq p$ $(i = 1, \ldots, n)$. Then, the probability that more than or equal to k events happen is at most $\frac{np}{k}$.*

Proof. Let $[n] = \{1, 2, \ldots, n\}$, $C_{n,t}$ be all size t subsets of $[n]$ (*e.g.,* $C_{3,2} = \{\{1,2\},\{1,3\},\{2,3\}\}$), and $C^i_{n,t}$ be all size t subsets of $[n]$ that contains i (*e.g.,* $C^1_{3,2} = \{\{1,2\},\{1,3\}\}$).

$$\begin{aligned}
k \cdot \Pr[\text{more than or equal to } k \text{ events happen}] &= k \cdot \sum_{t=k}^{n} \sum_{S \in C_{n,t}} \Pr[\bigwedge_{j \in S} A_j \wedge \bigwedge_{j \in [n] \setminus S} A_j^c] \\
&\leq \sum_{t=k}^{n} \sum_{S \in C_{n,t}} t \cdot \Pr[\bigwedge_{j \in S} A_j \wedge \bigwedge_{j \in [n] \setminus S} A_j^c] \\
&= \sum_{i=1}^{n} \sum_{t=k}^{n} \sum_{S \in C^i_{n,t}} \Pr[\bigwedge_{j \in S} A_j \wedge \bigwedge_{j \in [n] \setminus S} A_j^c] \\
&\leq \sum_{i=1}^{n} \sum_{t=1}^{n} \sum_{S \in C^i_{n,t}} \Pr[\bigwedge_{j \in S} A_j \wedge \bigwedge_{j \in [n] \setminus S} A_j^c] \\
&= \sum_{i=1}^{n} \Pr[A_i] \\
&\leq np
\end{aligned}$$

where ^{c} denotes complement. Therefore,

$$\Pr[\text{more than or equal to } k \text{ events happen}] \leq \frac{np}{k}$$

□

Proof of Theorem 1. We prove this by performing mathematical induction on the number of layers L.

base case: It is shown that the statement holds for $L = 1$.

By multivariate Hoeffding's inequality

$$\Pr[\|\hat{\boldsymbol{m}}_v^{(1)} - \boldsymbol{m}_v^{(1)}\|_2 \geq \frac{\varepsilon}{KB_{\text{op}}}] \leq \delta.$$

If $\|\hat{\boldsymbol{m}}_v^{(1)} - \boldsymbol{m}_v^{(1)}\|_2 < \frac{\varepsilon}{KB_{\text{op}}}$, then $\|\hat{\boldsymbol{z}}_v^{(1)} - \boldsymbol{z}_v^{(1)}\|_2 \leq K\|\hat{\boldsymbol{h}}_v^{(1)} - \boldsymbol{h}_v^{(1)}\|_2 \leq KB_{\text{op}}\|\hat{\boldsymbol{m}}_v^{(1)} - \boldsymbol{m}_v^{(1)}\|_2 < \varepsilon$. Here, we use $\|\boldsymbol{W}\boldsymbol{m}\|_2 \leq \|\boldsymbol{W}\|_{op}\|\boldsymbol{m}\|_2 \leq B_{op}\|\boldsymbol{m}\|$ (Assumption 1) and $\|\boldsymbol{z} - \boldsymbol{z}'\|_2 \leq K\|\boldsymbol{h} - \boldsymbol{h}'\|_2$ (Assumption 2), respectively. Therefore,

$$\Pr[\|\hat{\boldsymbol{z}}_v^{(1)} - \boldsymbol{z}_v^{(1)}\|_2 \geq \varepsilon] \leq \delta$$

inductive step: It is shown that the statement holds for $L = l+1$ if it holds for $L = l$. The inductive hypothesis is $\forall v \in V, \varepsilon > 0, \delta > 0, \Pr[\|\hat{\mathcal{O}}_z^{(l)}(v, \varepsilon, \delta) - \boldsymbol{z}_v\|_2 \geq \varepsilon] \leq \delta$.

Let $\bar{\boldsymbol{m}}_v^{(l+1)} = \frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \boldsymbol{z}_u^{(l+1)}$ ($r^{(l+1)} = |S^{(l+1)}|$), then by multivariate Hoeffding's inequality,

$$\Pr[\|\bar{\boldsymbol{m}}_v^{(l+1)} - \boldsymbol{m}_v^{(l+1)}\|_2 \geq \frac{\varepsilon}{2KB_{\text{op}}}] \leq \frac{\delta}{2}. \tag{1}$$

Let $\hat{\boldsymbol{z}}_u^{(l)} = \hat{\mathcal{O}}_z^{(l)}(u, \frac{\varepsilon}{4KB_{\text{op}}}, \frac{\varepsilon\delta}{16B_{\text{op}}KB_l})$ and $\kappa = \text{ceil}(\frac{r^{(l+1)}\varepsilon}{8KB_{\text{op}}B_l})$. If

$$\#\{u \in S^{(l+1)} \mid \|\hat{\boldsymbol{z}}_u^{(l)} - \boldsymbol{z}_u^{(l)}\|_2 \geq \frac{\varepsilon}{4KB_{\text{op}}}\} < \kappa \tag{2}$$

where $\#$ denotes the number of elements, then

$$\|\hat{\boldsymbol{m}}_v^{(l+1)} - \bar{\boldsymbol{m}}_v^{(l+1)}\|_2 = \|\frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} (\hat{\boldsymbol{z}}_u^{(l)} - \boldsymbol{z}_u^{(l)})\|_2$$

$$
\begin{aligned}
&\leq \frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \|\hat{\boldsymbol{z}}_u^{(l)} - \boldsymbol{z}_u^{(l)}\|_2 \\
&\leq \frac{1}{r^{(l+1)}} \left(r^{(l+1)} \cdot \frac{\varepsilon}{4KB_{\text{op}}} + (\kappa - 1) \cdot 2B_l \right) \\
&< \frac{\varepsilon}{4KB_{\text{op}}} + \frac{\varepsilon}{4KB_{\text{op}}} \\
&= \frac{\varepsilon}{2KB_{\text{op}}}
\end{aligned}
$$

Note that $\|\hat{\boldsymbol{z}}_u^{(l)}\|_2 \leq B_l$ also holds with the same argument with Lemma 10. Using a special case of the induction hypothesis (i.e., $\hat{\boldsymbol{z}}_u^{(l)} = \hat{\mathcal{O}}_z^{(l)}(u, \frac{\varepsilon}{4KB_{\text{op}}}, \frac{\varepsilon\delta}{16B_{\text{op}}KB_l})$):

$$
\Pr[\|\hat{\boldsymbol{z}}_u^{(l)} - \boldsymbol{z}_u^{(l)}\|_2 \geq \frac{\varepsilon}{4KB_{\text{op}}}] \leq \frac{\varepsilon\delta}{16B_{\text{op}}KB_l},
$$

the probability that (2) does not hold is

$$
\begin{aligned}
\Pr[\#\{u \in S^{(l+1)} \mid \|\hat{\boldsymbol{z}}_u^{(l)} - \boldsymbol{z}_u^{(l)}\|_2 \geq \frac{\varepsilon}{4KB_{\text{op}}}\} \geq \kappa] &\leq r^{(l+1)} \cdot \Pr[\|\hat{\boldsymbol{z}}_u^{(l)} - \boldsymbol{z}_u^{(l)}\|_2 \geq \frac{\varepsilon}{4KB_{\text{op}}}] \cdot \frac{1}{\kappa} \\
&\leq r^{(l+1)} \cdot \frac{\varepsilon\delta}{16B_{\text{op}}KB_l} \cdot \frac{8KB_{\text{op}}B_l}{r^{(l+1)}\varepsilon} \\
&= \frac{\delta}{2}
\end{aligned}
$$

Here, we use Lemma 13 to obtain the first inequality. Therefore,

$$
\Pr[\|\hat{\boldsymbol{m}}_v^{(l+1)} - \bar{\boldsymbol{m}}_v^{(l+1)}\|_2 \geq \frac{\varepsilon}{2KB_{\text{op}}}] \leq \frac{\delta}{2} \tag{3}
$$

Combining (1) and (3), using the triangle inequality,

$$
\Pr[\|\hat{\boldsymbol{m}}_v^{(l+1)} - \boldsymbol{m}_v^{(l+1)}\|_2 \geq \frac{\varepsilon}{KB_{\text{op}}}] \leq \delta
$$

If $\|\hat{\boldsymbol{m}}_v^{(l+1)} - \boldsymbol{m}_v^{(l+1)}\|_2 < \frac{\varepsilon}{KB_{\text{op}}}$, then $\|\hat{\boldsymbol{z}}_v^{(l+1)} - \boldsymbol{z}_v^{(l+1)}\|_2 \leq K\|\hat{\boldsymbol{h}}_v^{(l+1)} - \boldsymbol{h}_v^{(l+1)}\|_2 \leq KB_{\text{op}}\|\hat{\boldsymbol{m}}_v^{(l+1)} - \boldsymbol{m}_v^{(l+1)}\|_2 < \varepsilon$. Therefore,

$$
\Pr[\|\hat{\boldsymbol{z}}_v^{(l+1)} - \boldsymbol{z}_v^{(l+1)}\|_2 \geq \varepsilon] \leq \delta
$$

□

Proof of Theorem 2. We prove this by performing mathematical induction on the number of layers.

base case: It is shown that the statement holds for $L = 1$.

Algorithm 2 asks one query to $\mathcal{O}_{\text{deg}}$, $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ queries to $\mathcal{O}_G$, and $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ queries to $\mathcal{O}_{\text{feature}}$. Therefore, the query complexity is $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ in total.

inductive step: It is shown that the statement holds for $L = l + 1$ if it holds for $L = l$.

Algorithm 3 asks one query to $\mathcal{O}_{\text{deg}}$, $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ queries to $\mathcal{O}_G$, and $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ queries to $\hat{\mathcal{O}}_z^{(l)}(u, \Theta(\varepsilon), \Theta(\varepsilon\delta))$. Using the induction hypothesis, the query complexity of $\hat{\mathcal{O}}_z^{(l)}(u, \Theta(\varepsilon), \Theta(\varepsilon\delta))$ is $O(\frac{1}{\varepsilon^{2l}}(\log\frac{1}{\varepsilon} + \log\frac{1}{\varepsilon\delta})^{l-1}(\log\frac{1}{\varepsilon\delta})) = O(\frac{1}{\varepsilon^{2l}}(\log\frac{1}{\varepsilon} + \log\frac{1}{\delta})^l)$. Therefore, the query complexity is $O(\frac{1}{\varepsilon^{2(l+1)}}(\log\frac{1}{\varepsilon} + \log\frac{1}{\delta})^l(\log\frac{1}{\delta}))$ in total □

Lemma 14. *Let* $H_k = K'B_{op}B_{k-1} + \|\sigma'(\mathbf{0}_{d_l})\|_2$, $G_l^{(l)} = H_lB_{l-1}$, *and* $G_l^{(k+1)} = H_{k+1}B_FG_l^{(k)}$ $(k \geq l)$. *Under Assumptions 1, 2, and 3,* $\|\sigma'(\boldsymbol{h}_v^{(l)})\|_2$ *and* $\|\frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F$ $(k \geq l)$ *are bounded by* H_k *and* $G_l^{(k)}$, *respectively.*

Proof of Lemma 14.

$$\|\sigma'(\boldsymbol{h}_v^{(k)})\|_2 \leq K'\|\boldsymbol{h}_v^{(0)}\|_2 + \|\sigma'(\mathbf{0}_{d_k})\|_2 \leq K' B_{\text{op}} B_{k-1} + \|\sigma'(\mathbf{0}_{d_k})\|_2 = H_k$$

$$\|\frac{\partial \boldsymbol{z}_v^{(l)}}{\partial \boldsymbol{W}^{(l)}}\|_F = \|\sigma'(\boldsymbol{h}_v^{(l)})\boldsymbol{m}_v^{(l)T}\|_F \leq H_l B_{l-1} = G_l^{(l)}$$

$$\begin{aligned}
\|\frac{\partial \boldsymbol{z}_v^{(k+1)}}{\partial \boldsymbol{W}^{(l)}}\|_F &= \|\frac{\partial \boldsymbol{z}_v^{(k+1)}}{\partial \boldsymbol{h}_v^{(k+1)}} \frac{\partial \boldsymbol{h}_v^{(k+1)}}{\partial \boldsymbol{m}_v^{(k+1)}} \sum_{u \in \mathcal{N}(v)} \frac{\partial \boldsymbol{m}_v^{(k+1)}}{\partial \boldsymbol{z}_v^{(k)}} \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F \\
&\leq \|\frac{\partial \boldsymbol{z}_v^{(k+1)}}{\partial \boldsymbol{h}_v^{(k+1)}}\|_F \|\boldsymbol{W}^{(k+1)}\|_F \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} \|\frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F \\
&\leq H_{k+1} B_F G_l^{(k)} = G_l^{(k+1)}
\end{aligned}$$

□

Proof of Theorem 3. Let $E_l^{(k)}$ and $D_l^{(k)}$ $(k \geq l)$ be

$$\begin{aligned}
E_l^{(l)} &= \frac{1}{K}(\frac{H_l}{B_{\text{op}}} + B_{l-1}K'), \\
E_l^{(k+1)} &= \frac{G_l^{(k)}\sqrt{d_{l-1}d_l}H_{k+1}B_F}{2B_k B_{\text{op}} K} + \frac{B_F}{K}(\frac{E_l^{(k)}H_{k+1}}{2B_{\text{op}}} + K'G_l^{(k)}), \\
D_l^{(l)} &= 1, \\
D_l^{(k+1)} &= \frac{d_{l-1}d_l}{2} + \frac{D_l^{(k)}G_l^{(k)}}{2E_l^{(k)}B_k} + 1.
\end{aligned}$$

They are constants with respect to the input size, ε, and δ. We prove $C = \frac{1}{E_l^{(L)}}$ and $D = \frac{1}{D_l^{(L)}}$.

We prove this by performing mathematical induction on the number of layers L.

base case: It is shown that the statement holds for $L = l$.

From the proof of Theorem 1,

$$\Pr[\|\hat{\boldsymbol{m}}_v^{(l)} - \boldsymbol{m}_v^{(l)}\|_2 \geq \frac{\varepsilon}{KB_{\text{op}}}] \leq \delta$$

. If $\|\hat{\boldsymbol{m}}_v^{(l)} - \boldsymbol{m}_v^{(l)}\|_2 < \frac{\varepsilon}{KB_{\text{op}}}$,

$$\begin{aligned}
\|\widehat{\frac{\partial \boldsymbol{z}_v^{(l)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_v^{(l)}}{\partial \boldsymbol{W}^{(l)}}\|_F &= \|\sigma'(\hat{\boldsymbol{h}}_v^{(l)})\hat{\boldsymbol{m}}_v^{(l)T} - \sigma'(\boldsymbol{h}_v^{(l)})\boldsymbol{m}_v^{(l)T}\|_F \\
&\leq \|\sigma'(\hat{\boldsymbol{h}}_v^{(l)})\|_2 \|\hat{\boldsymbol{m}}_v^{(l)} - \boldsymbol{m}_v^{(l)}\|_2 + \|\boldsymbol{m}_v^{(l)}\|_2 \|\sigma'(\hat{\boldsymbol{h}}_v^{(l)}) - \sigma'(\boldsymbol{h}_v^{(l)})\|_2 \\
&< H_l \frac{\varepsilon}{KB_{\text{op}}} + B_{l-1}K'\frac{\varepsilon}{K} \\
&= \frac{1}{K}(\frac{H_l}{B_{\text{op}}} + B_{l-1}K')\varepsilon = E_l^{(l)}\varepsilon
\end{aligned}$$

Therefore,

$$\Pr[\|\widehat{\frac{\partial \boldsymbol{z}_v^{(l)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_v^{(l)}}{\partial \boldsymbol{W}^{(l)}}\|_F \geq E_l^{(l)}\varepsilon] \leq D_l^{(l)}\delta = \delta$$

inductive step: It is shown that the statement holds for $L = k + 1$ if it holds for $L = k$.

Let

$$\overline{\frac{\partial \boldsymbol{z}^{(k+1)}}{\partial \boldsymbol{W}^{(l)}}} = \frac{\partial \boldsymbol{z}_v^{(k+1)}}{\partial \boldsymbol{h}_v^{(k+1)}} \frac{\partial \boldsymbol{h}_v^{(k+1)}}{\partial \boldsymbol{m}_v^{(k+1)}} \frac{1}{r^{(k+1)}} \sum_{u \in S^{(k+1)}} \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}$$

$$E_{\text{sample}} = \| \frac{1}{r^{(k+1)}} \sum_{u \in S^{(k+1)}} \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}} - \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F$$

By Multivariate Hoeffding's inequality,

$$\begin{aligned}
&\Pr[E_{\text{sample}} \geq \frac{G_l^{(k)} \sqrt{d_{l-1} d_l}}{2 B_k B_{\text{op}} K} \varepsilon] \\
&\leq 2 d_{l-1} d_l d_k \exp(-\frac{1}{2 G_l^{(k)2} d_{l-1} d_l d_k} \frac{G_l^{(k)2} d_{l-1} d_l}{4 B_k^2 B_{\text{op}}^2 K^2} \varepsilon^2 \frac{8 B_k^2 B_{\text{op}}^2 K^2 d_k}{\varepsilon^2} \log(\frac{4 d_k}{\delta})) \\
&= \frac{d_{l-1} d_l}{2} \delta
\end{aligned}$$

If $E_{\text{sample}} < \frac{G_l^{(k)} \sqrt{d_{l-1} d_l}}{2 B_k B_{\text{op}} K} \varepsilon$, then

$$\begin{aligned}
\| \overline{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F &\leq \| \frac{\partial \boldsymbol{z}_v^{(k+1)}}{\partial \boldsymbol{h}_v^{(k+1)}} \|_F \| \frac{\partial \boldsymbol{h}_v^{(k+1)}}{\partial \boldsymbol{m}_v^{(k+1)}} \|_F E_{\text{sample}} \\
&< \frac{G_l^{(k)} \sqrt{d_{l-1} d_l} H_{k+1} B_F}{2 B_k B_{\text{op}} K} \varepsilon
\end{aligned}$$

Therefore,

$$\Pr[\| \overline{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F \geq \frac{G_l^{(k)} \sqrt{d_{l-1} d_l} H_{k+1} B_F}{2 B_k B_{\text{op}} K} \varepsilon] \leq \frac{d_{l-1} d_l}{2} \delta \tag{4}$$

Let $\widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}}$ be the gradient of $\hat{\boldsymbol{z}}_u^{(k)}$, which is obtained by $\hat{\mathcal{O}}_z^{(k)}(u, \frac{\varepsilon}{4 K B_{\text{op}}}, \frac{\varepsilon \delta}{16 B_{\text{op}} K B_{l-1}})$, with respect to $\boldsymbol{W}^{(l)}$. Let $\kappa = \text{ceil}(\frac{r^{(k+1)} E_l^{(k)} \varepsilon}{8 K B_{\text{op}} G_l^{(k)}})$. If

$$\#\{u \in S^{(l+1)} \mid \| \widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F \geq \frac{E_l^{(k)} \varepsilon}{4 K B_{\text{op}}} \} < \kappa \tag{5}$$

where $\#$ denotes the number of elements, then

$$\begin{aligned}
\frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \| \widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F &\leq \frac{1}{r^{(l+1)}} (r^{(l+1)} \cdot \frac{E_l^{(k)} \varepsilon}{4 K B_{\text{op}}} + (\kappa - 1) \cdot 2 G_l^{(k)}) \\
&< \frac{E_l^{(k)} \varepsilon}{4 K B_{\text{op}}} + \frac{E_l^{(k)} \varepsilon}{4 K B_{\text{op}}} \\
&= \frac{E_l^{(k)} \varepsilon}{2 K B_{\text{op}}}
\end{aligned}$$

Note that $\| \widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} \| \leq G_l^{(k)}$ also holds with the same argument with Lemma 14. Using the induction hypothesis,

$$\Pr[\| \widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F \geq \frac{E_l^{(k)} \varepsilon}{4 K B_{\text{op}}}] \leq \frac{D_l^{(k)} \varepsilon \delta}{16 B_{\text{op}} K B_k}$$

Therefore, the probability that (5) does not hold is

$$\Pr[\#\{u \in S^{(l+1)} \mid \| \widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}} \|_F \geq \frac{E_l^{(k)} \varepsilon}{4 K B_{\text{op}}} \} \geq \kappa]$$

$$
\begin{aligned}
&\leq r^{(l+1)} \cdot \Pr[\|\widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F \geq \frac{E_l^{(k)}\varepsilon}{4KB_{\text{op}}}] \cdot \frac{1}{\kappa} \\
&\leq r^{(l+1)} \cdot \frac{D_l^{(k)}\varepsilon\delta}{16B_{\text{op}}KB_k} \cdot \frac{8KB_{\text{op}}G_l^{(k)}}{r^{(k+1)}E_l^{(k)}\varepsilon} \\
&= \frac{D_l^{(k)}G_l^{(k)}}{2E_l^{(k)}B_k}\delta
\end{aligned}
$$

Here, we use Lemma 13 to obtain the first inequality. Therefore,

$$
\Pr[\frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \|\widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}})\|_F \geq \frac{E_l^{(k)}\varepsilon}{2KB_{\text{op}}}] \leq \frac{D_l^{(k)}G_l^{(k)}\delta}{2E_l^{(k)}B_k} \tag{6}
$$

From the proof of Theorem 1,

$$
\Pr[\|\hat{\boldsymbol{h}}_v^{(l+1)} - \boldsymbol{h}_v^{(l+1)}\|_2 \geq \frac{\varepsilon}{K}] \leq \delta \tag{7}
$$

If $\frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \|\widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F < \frac{E_l^{(k)}\varepsilon}{2KB_{\text{op}}}$ and $\|\hat{\boldsymbol{h}}_v^{(l+1)} - \boldsymbol{h}_v^{(l+1)}\|_2 < \frac{\varepsilon}{K}$, then

$$
\begin{aligned}
&\|\widehat{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}} - \overline{\frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}}\|_F \\
&\leq \|\sigma'(\hat{\boldsymbol{h}}_v^{(k+1)})\|_2 \|\boldsymbol{W}^{(k+1)}\|_F \frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \|\widehat{\frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F \\
&\quad + \|\sigma'(\hat{\boldsymbol{h}}_v^{(k+1)}) - \sigma'(\boldsymbol{h}_v^{(k+1)})\|_2 \|\boldsymbol{W}^{(k+1)}\|_F \|\frac{1}{r^{(l+1)}} \sum_{u \in S^{(l+1)}} \frac{\partial \boldsymbol{z}_u^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F \\
&< H_{k+1}B_F \frac{E_l^{(k)}\varepsilon}{2KB_{\text{op}}} + K' \frac{\varepsilon}{K} B_F G_l^{(k)} \\
&= \frac{B_F}{K}(\frac{E_l^{(k)}H_{k+1}}{2B_{\text{op}}} + K'G_l^{(k)})\varepsilon
\end{aligned}
$$

Therefore, from (6) and (7),

$$
\Pr[\|\widehat{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}} - \overline{\frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}}\|_F \geq \frac{B_F}{K}(\frac{E_l^{(k)}H_{k+1}}{2B_{\text{op}}} + K'G_l^{(k)})\varepsilon] \leq \frac{D_l^{(k)}G_l^{(k)}}{2E_l^{(k)}B_k}\delta + \delta \tag{8}
$$

Combining (4) and (8), using the triangle inequality,

$$
\Pr[\|\widehat{\frac{\partial \boldsymbol{z}_v^{(L)}}{\partial \boldsymbol{W}^{(l)}}} - \frac{\partial \boldsymbol{z}_v^{(k)}}{\partial \boldsymbol{W}^{(l)}}\|_F \geq E_l^{(k+1)}] \frac{d_{l-1}d_l}{2}\delta + \frac{D_l^{(k)}G_l^{(k)}}{2E_l^{(k)}B_k}\delta + \delta = D_l^{(k+1)}\delta
$$

□

Proof of Theorem 5. There exists $a \in \mathbb{R}$ such that $\sigma(a) \neq \sigma(0)$ because σ is not a constant. Consider 1-layer GraphSAGE-GCN and the following two types of inputs:

- G is the clique K_n, $\boldsymbol{W}^{(1)} = 1$, and $\boldsymbol{x}_i = 0$ for all nodes $i \in V$.
- G is the clique K_n, $\boldsymbol{W}^{(1)} = 1$, $\boldsymbol{x}_i = 0 (i \neq v)$ for some node $v \in V$, and $\boldsymbol{x}_v = an$.

Then, for the former input, $\boldsymbol{z}_i^{(1)} = \sigma(0)$ for all $i \in V$. For the latter type of inputs, $\boldsymbol{z}_i^{(1)} = \sigma(a)$ for all $i \in V$. If we set $\varepsilon = |\sigma(a) - \sigma(0)|/2$, the algorithm has to distinguish the types of inputs with high probability. However, it needs $\Omega(n)$ queries to $\mathcal{O}_{\text{feature}}$ to find the node v. As for $\boldsymbol{W}$, we set $\boldsymbol{W}^{(1)} = an$ and $\boldsymbol{x}_v = 1$. Then the same argument follows. □

Proof of Theorem 6. Consider the following 1-layer GraphSAGE-GCN

$$\boldsymbol{h}_v \leftarrow \textsc{ReLU}(\boldsymbol{W} \cdot \textsc{MEAN}(\{\boldsymbol{x}_u \mid u \in \mathcal{N}(v)\}))$$
$$\boldsymbol{z}_v \leftarrow \boldsymbol{h}_v / \|\boldsymbol{h}_v\|_2$$

and the following two types of inputs:

- G is the clique K_n, $\boldsymbol{W}$ is the identity matrix $\boldsymbol{I}_2$, $\boldsymbol{x}_i = [0, 0]^\top (i \neq v)$ for some node $v \in V$, and $\boldsymbol{x}_v = [1, 0]^\top$.
- G is the clique K_n, $\boldsymbol{W}$ is the identity matrix $\boldsymbol{I}_2$, $\boldsymbol{x}_i = [0, 0]^\top (i \neq v)$ for some node $v \in V$, and $\boldsymbol{x}_v = [0, 1]^\top$.

Then, for the former type of inputs, $\boldsymbol{h}_i = [1/n, 0]^\top$, $\boldsymbol{z}_i = [1, 0]^\top$, and $\frac{\partial z_{i2}}{\partial W_{21}} = 1$ for all $i \in V$. For the latter type of inputs, $\boldsymbol{h}_i = [0, 1/n]^\top$, $\boldsymbol{z}_i = [0, 1]^\top$, and $\frac{\partial \boldsymbol{z}_{i2}}{\partial W_{21}} = 0$ for all $i \in V$. If we set $\varepsilon = 1/2$, the algorithm has to distinguish the types of inputs with high probability, both for the inference and gradient. However, it needs $\Omega(n)$ queries to $\mathcal{O}_{\text{feature}}$ to find the node v. □

Proof of Theorem 7. Consider the following 1-layer GraphSAGE-GCN

$$\boldsymbol{h}_v \leftarrow \boldsymbol{W} \cdot \textsc{MEAN}(\{\boldsymbol{x}_u \mid u \in \mathcal{N}(v)\}))$$
$$\boldsymbol{z}_v = \textsc{ReLU}(\boldsymbol{h}_v)$$

and consider the following two types of inputs:

- G is the clique K_n, $\boldsymbol{W} = [-1, 1]$, $\boldsymbol{x}_i = [1, 1]^\top (i \neq v)$ for some node $v \in V$, and $\boldsymbol{x}_v = [1, 2]^\top$.
- G is the clique K_n, $\boldsymbol{W} = [-1, 1]$, $\boldsymbol{x}_i = [1, 1]^\top (i \neq v)$ for some node $v \in V$, and $\boldsymbol{x}_v = [1, 0]^\top$.

Then, for the former type of inputs, $\textsc{MEAN}(\{\boldsymbol{x}_u \mid u \in \mathcal{N}(v)\}) = [1, 1 + \frac{1}{n}]^\top$, $\boldsymbol{h}_v = \boldsymbol{z}_v = \frac{1}{n}$, and $\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}} = [1, 1 + \frac{1}{n}]$ for all $i \in V$. For the latter type of inputs, $\textsc{MEAN}(\{\boldsymbol{x}_u \mid u \in \mathcal{N}(v)\}) = [1, 1 - \frac{1}{n}]^\top$, $\boldsymbol{h}_v = -\frac{1}{n}$, $\boldsymbol{z}_v = 0$, and $\frac{\partial \boldsymbol{z}_v}{\partial \boldsymbol{W}} = [0, 0]$ for all $i \in V$. If we set $\varepsilon = 1/2$, the algorithm has to distinguish the types of inputs with high probability. However, it needs $\Omega(n)$ queries to $\mathcal{O}_{\text{feature}}$ to find the node v. □

Proof of Theorem 8. Consider the following 1-layer GraphSAGE-pool

$$\boldsymbol{h}_v \leftarrow \sigma(\boldsymbol{W} \cdot \max(\{\boldsymbol{x}_u \mid u \in \mathcal{N}(v)\}))$$

and the following two types of inputs:

- G is the clique K_n, $\boldsymbol{W} = 1$, and $\boldsymbol{x}_i = 0$ for all nodes $v \in V$.
- G is the clique K_n, $\boldsymbol{W} = 1$, $\boldsymbol{x}_i = 0$ $(i \neq v)$ for some node $v \in V$, and $\boldsymbol{x}_v = 1$.

Then, for the former type of inputs, $\boldsymbol{z}_i = \sigma(0)$ for all $i \in V$. For the latter type of inputs, $\boldsymbol{z}_i = \sigma(1)$ for all $i \in V$. If we set $\varepsilon = |\sigma(1) - \sigma(0)|/2$, the algorithm has to distinguish the types of inputs with high probability. However, it needs $\Omega(n)$ queries to $\mathcal{O}_{\text{feature}}$ to find the node v. □

Proof of Lemma 9. Consider the following 1-layer GCN

$$z = \sigma(\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\boldsymbol{X}\boldsymbol{W})$$

and the following two types of inputs:

- G is a star where $v \in V$ is the center of G, $\boldsymbol{W} = 1$, and all features are 0.
- G is a star where $v \in V$ is the center of G, $\boldsymbol{W} = 1$, and features of $\sqrt{2n}$ leafs are 1 and the features of other nodes are 0.

Then, for the former type of inputs, $\boldsymbol{z}_v = \sigma(0)$. For the latter type of inputs, $\boldsymbol{z}_v = \sigma(1)$. Suppose there is a constant time algorithm that calculates $\boldsymbol{z}_v$ in arbitrary precision with arbitrary probability in constant time. If we set $\varepsilon = (\sigma(1) - \sigma(0))/2$ and $\delta = 1/3$, this algorithm has to distinguish the types of inputs with probability $2/3$ within C accesses. Suppose the input is of the latter type. When this algorithm accesses $\mathcal{O}_{\text{feature}}$, the probability that all the returned values are 0 is at least $(\frac{n-1-\sqrt{2n}}{n-1})^C$. This probability goes to 1 when n goes to ∞. Therefore, if the size of the graph is sufficiently large, this algorithm cannot distinguish the type of input with probability $2/3$. $\square$