Graph Explicit Neural Networks: Explicitly Encoding Graphs for Efficient and Accurate Inference

Yiwei Wang
National University of Singapore
Singapore
wangyw_seu@foxmail.com

Bryan Hooi
National University of Singapore
Singapore
bhooi@comp.nus.edu.sg

Yozen Liu
Snap Inc.
The United States of America
yliu2@snap.com

Neil Shah
Snap Inc.
The United States of America
nshah@snap.com

ABSTRACT

As the state-of-the-art graph learning models, the message passing based neural networks (MPNNs) **implicitly** use the graph topology as the "pathways" to propagate node features. This implicit use of graph topology induces the MPNNs' over-reliance on (node) features and high inference latency, which hinders their large-scale applications in industrial contexts. To mitigate these weaknesses, we propose the **Graph Explicit Neural Network (GENN)** framework. GENN can be flexibly applied to various MPNNs and improves them by providing more efficient and accurate inference that is robust in feature-constrained settings. Specifically, we carefully incorporate recent developments in network embedding methods to efficiently prioritize the graph topology for inference. From this vantage, GENN explicitly encodes the topology as an important source of information to mitigate the reliance on node features. Moreover, by adopting knowledge distillation (KD) techniques, GENN takes an MPNN as the *teacher* to supervise the training for better effectiveness while avoiding the teacher's high inference latency. Empirical results show that our GENN infers dramatically faster than its MPNN teacher by 40×-78×. In terms of accuracy, GENN yields significant gains (more than 40%) for its MPNN teacher when the node features are limited based on our explicit encoding. Moreover, GENN outperforms the MPNN teacher even in feature-rich settings thanks to our KD design.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Supervised learning by classification**.

KEYWORDS

graph neural networks, knowledge distillation, explicit encoding

WSDM '23, February 27-March 3, 2023, Singapore, Singapore
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570388

ACM Reference Format:
Yiwei Wang, Bryan Hooi, Yozen Liu, and Neil Shah. 2023. Graph Explicit Neural Networks: Explicitly Encoding Graphs for Efficient and Accurate Inference. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570388

1 INTRODUCTION

Inference efficiency and accuracy are two popular criteria to evaluate a machine learning system. Nowadays, growing demands of graph learning applications in the industry necessitate models which infer with low latency [9, 28, 50]. Besides, the increasing privacy-preserving services pose accuracy challenges to graph learning systems under limited feature constraints [26, 30, 38]. Yet, state-of-the-art graph learning models: message passing neural networks (MPNNs) [16, 24, 41], struggle in these latency- and feature-constrained settings [8, 50].

Graph topology and (node) features are two main sources of information for graph learning. The former is irregular and discrete, while the latter is regular and continuous, making the latter more conveniently encoded by neural networks [19]. Given these data characteristics, MPNNs explicitly encode the node features but **implicitly** uses the graph topology as "pathways" to propagate node features. We argue that the way MPNNs implicitly use graph topology creates these seemingly disparate, but ultimately coupled, fundamental limitations: over-reliance on node features [8] and slow inference [50]. These disadvantages reduce the applicability of MPNNs for inference in large-scale industry applications:

Overreliance on Features [8]: Because the graph topology is only implicitly used by MPNNs as "pathways" for fetching the neighbors' features, if the node features are not informative, graph topology has no valuable information to convey. As we show in Fig. 1 (blue line), the low-feature settings severely degrade MPNN performance, and hurt their practicality [8, 33]. This phenomenon is consistent across various graph benchmarks (See Sec. 4.2) and implies that MPNNs rely on the informative node features for accurate predictions. In many industrial services, the models only have limited access to node features for reasons of privacy, fairness, or availability [27, 29, 39], where overreliance on features can lead to serious degradation of effectiveness.

High inference latency [50]: The implicit use of graph topology forces MPNNs to propagate the features over the edges; in other words, MPNNs have to fetch and transform the high dimensional representations of many neighbors to expand the receptive fields [44] on the graph topology. As a result, the prediction on a single node of MPNNs requires many neighbors' feature look-ups as well as Multiplication-and-ACcumulation (MAC) operations and thus causes high inference latency [50]. This makes the MPNNs challenging to deploy for latency-constrained applications which require fast inference. This issue is also known as neighbor explosion [50] when the number of layers is high (see Fig. 1).

Because MPNNs' implicit use of graph topology induces both these issues, we aim to **explicitly** utilize the information from graph topology to address them and improve MPNNs. However, graph topology is irregular and discrete, which makes it nontrivial to handle with an MLP. To process the graph topology, we utilize the recently developed method InstantEmbedding [33], from a classical domain of graph learning: Network Embeddings (NEs) [37, 40] to transform the irregular graph topology into regular node-level embeddings. InstantEmbedding is efficient and inductive, which forms our basis for quickly prioritizing the graph topology for inference. On the other hand, when the node features are rich, MPNNs are still the most effective graph learning models. Therefore, in our work, we propose a knowledge distillation (KD) based GNN framework: **Graph Explicit Neural Network (GENN)** to improve MPNNs for the efficient and accurate inference that is robust to the feature-constrained settings. GENN takes an MPNN model as the teacher model during training, but avoids using it for inference, so that we can benefit from the distilled knowledge of MPNNs while avoiding their high inference latency.

We use the MPNN teacher to supervise InstantEmbedding as the student model for fast inference and enjoy the enhanced effectiveness from the teacher. However, InstantEmbedding is unparameterized and thus cannot be supervised by MPNNs directly. To transform the output of InstantEmbedding in the direction that adapts to the MPNN teachers, we propose the *topology student* module that constructs a multi-layer perceptron (MLP) upon InstantEmbedding. The distilled *topology student* augments the InstantEmbedding's outputs for higher effectiveness. In this way, we explicitly encode the graph topology as an important source of information for inference, so as to mitigate the overreliance on the node features.

Given our *topology student* that explicitly encodes graph topology, there still exists the possibility for higher effectiveness by utilizing the node features. Therefore, we propose an MLP based *feature student* module to encode the input node features under the guidance of the MPNN teacher as well. To fuse the information in graph topology and features, one practical obstacle is that the correlation between the graph data and the ground truth labels is very complex: the labels can be more correlated with either graph topology or features [10, 43] depending on the instance. In order to extract the most relevant information from graph topology and features, we utilize an attention based fusion head to learn importance weights for our student modules, so as to fuse the graph topology and features in an adaptive and interpretable manner [43].

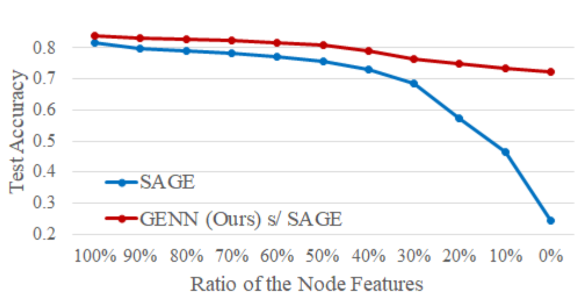
During inference, GENN makes the predictions by fusing the outputs of two student modules, avoiding any neighbor feature fetching or transformation, unlike MPNNs. The inference of GENN only includes two efficient students and a fusion head, which significantly improves inference efficiency over the complex MPNN teachers (see Fig. 1 right). In addition, GENN explicitly encodes the graph topology as an important source of information, which guarantees its effectiveness even with limited node features (see Fig. 1 left). Furthermore, our GENN has natural interpretability, by modulating relative importance of graph topology and features' contributions to predictions via attention weights learned by our fusion head (see Fig. 3).

We evaluate our GENN on transductive and inductive node classification tasks using multiple standard node classification benchmark datasets. Our results show that GENN is more effective (accurate) than the MPNN teacher given rich node features, and significantly outperforms the teacher when the node features are limited (by more than 40% on many datasets). Qualitatively, GENN adaptively assigns appropriate attention weights to the semantic embeddings in different datasets. Regarding inference efficiency, GENN enjoys 40×-78× faster inference than the MPNN teacher, greatly improving the deployability prospects of such a model. These results suggest that our GENN is a better choice for accurate and fast inference in graph learning, especially for latency- or feature-constrained applications.

2 RELATED WORK

Graph Neural Networks (GNNs). Early GNNs generalize convolutions to graphs [2, 7] and are later simplified to message passing (MP) neural networks (MPNNs) – most modern GNNs proposed afterwards can be seen as MPNNs, with architectural differences. For example, GAT employs attention [41], and GCNII employs residual connections [4]. Unlike these works, our work proposes a new inference paradigm to improve over MPNNs. We do not implicitly utilize the graph topology as "pathways" to transmit the node features, but explicitly encode the graph topology as an essential source of information for accurate and efficient inference on the graph data. For any MPNN model, our GENN takes it as a teacher and enhances its inference performance not only on efficiency but also on the effectiveness no matter whether the input node features are informative or not.

Inference Acceleration. Inference acceleration improvements largely build on hardware advances [23] and algorithmic advances through pruning [17] and quantization [15]. For GNNs, pruning [53] and quantizing parameters [52] have been studied. These approaches speed up GNN inference to a certain extent, but do not eliminate the core overhead of MP owing to neighbor data dependency. Concurrently, Graph-MLP also tries to bypass GNN neighbor fetching [21] by training an MLP with a neighbor-aware contrastive loss, but it only considers transductive settings, and not the more practical inductive setting. GLNN [50] proposes knowledge distillation of a teacher GNN model into a student MLP, which speeds up inference by avoiding MP, but does not utilize the graph topology during inference. Our method resolves these issues by explicitly encoding and adaptively fusing the information in graph topology. Note that several works also focus on speeding up GNN during *training* [5, 45, 54], which are complementary to our goal on speeding up model *inference*.



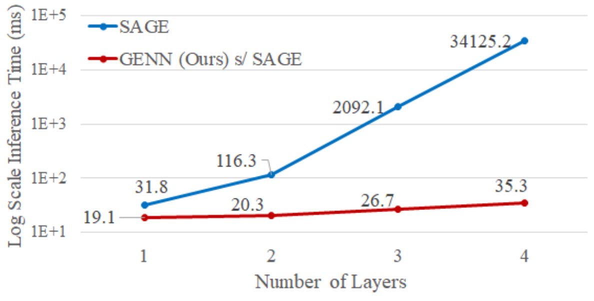


Figure 1: Under the supervision of the MPNN teacher SAGE [16], our GENN achieves more accurate and efficient inference than the teacher ("s/" denotes "supervised by"). (*left*) The test accuracy (in y-axis) of SAGE and our GENN on node classification of the Cora dataset [47] with different ratios (x-axis) of the accessible node features in terms of the input channels. (*right*) The inference time for ten random nodes in the OGB-Products dataset [20].

Network Embeddings (NE). Historically, the graph learning community developed methods which view learning NEs as a dimension reduction problem, where the goal is to preserve distances between nodes in a low-dimensional manifold [37, 40]. Some work proposes neural methods for NEs [14, 32] based on random walks. Later yet, [34] unified several skip-gram and factorization-based NE approaches under a single paradigm. Some recent works such as RandNE [51] and FastRP [3] iteratively project the adjacency matrix to simulate higher-order interactions between nodes. [33] proposes a local NE method based on Personalized PageRank that can scale to large graphs. Notably, these recent NE methods exhibit inductiveness and promising efficiency.

3 METHODOLOGY

As calls for privacy-preserving ML increase [38], the graph learning services are facing increasing challenges on the accuracy given limited features. For example, since the pivotal iOS 14.5 update [30], leading social network companies are estimated to bear a loss of $10bn due to the restricted use of features [26]. To retain strong user experience and revenue numbers, it is therefore desirable to deploy effective graph learning models which are robust to feature-constrained settings. In addition, the growing graph learning services require the models to infer with low latency [9, 28, 50]. However, as we have analyzed in Sec. 1, the state-of-the-art MPNN graph learning models [26, 30, 38], struggle in these feature- and latency- constrained settings. Specifically, the way that MPNNs implicitly use the graph topology creates problems of over-reliance on node features [8] and slow inference [50].

To overcome these disadvantages of MPNNs so as to improve their applicability in the industrial contexts, we propose the Graph-Explicit Neural Network (GENN). GENN is a knowledge distillation based GNN framework that explicitly encodes the graph topology for efficient and accurate inference. To benefit from the suprior effectiveness of MPNNs given rich features, GENN takes an MPNN as the teacher model, and introduces two student modules, learning from graph topology and features respectively, plus an attention-based fusion mechanism. The two student modules explicitly encode graph topology and features as semantic embeddings, while the attention-based mechanism adaptively extracts the valuable information from students. Both student modules are supervised by the MPNN teacher during training, and together offer efficient and accurate inference. While these technical components are straightforward, they have extremely strong practical implications when used together carefully. Fig. 2 illustrates the classical MPNNs and our GENN framework, which we introduce in detail next.

MPNN Teacher. We take an MPNN as the teacher model to produce pseudo-labels $\mathbf{y}_i^{teacher}$:

$$\mathbf{y}_i^{\text{teacher}} = \text{MPNN}(\mathcal{G}, i, \{\mathbf{x}_i, i \in \mathcal{V}\}) \tag{1}$$

where i is the node index, $\mathbf{x}_i$ is the input node feature of node i, $\mathcal{G}$ is the graph topology, $\mathcal{V}$ is the set of nodes, and $\mathbf{y}_{i,\text{teacher}} \in \mathbb{R}^C$ is the predicted logit vector taking the input of only the node features, where C is the number of classes. Since the MPNN teacher needs to operate the message passing to fetch the features of the neighbors multiple hops away from the target node i, it takes the set of node features $\{\mathbf{x}_i, i \in \mathcal{V}\}$ as the input instead of $\mathbf{x}_i$ itself.

Topology Student. Graph topology is an essential source of information for graph learning. When the node features are limited, the graph topology is especially crucial to be explicitly encoded as an independent source of information. Nowadays, more and more industrial applications of graph learning, such as social network analysis, are facing the restrictions on the utilization of node features due to the issues of privacy, fairness, or availability [27, 29, 39]. In this sense, the graph learning methods are desired to utilize the graph topology better to guarantee their effectiveness given the limited features.

However, graph topology is irregular and discrete, which makes it difficult to be directly processed by a MLP. To encode the graph topology, we review a classical kind of graph learning methods: network embeddings (NEs) [37, 40]. In contrast to MPNNs, the recent inductive NE method InstantEmbedding [33] is efficient to transform the graph topology into node-wise representations, and scales to large graphs, but is less effective on graph learning than MPNNs. Therefore, we try to take InstantEmbedding as an efficient student to learn from the more effective MPNN teachers.

Because InstantEmbedding is unparameterized and cannot be supervised by the MPNN teacher directly, we build an MLP module to transform the outputs of InstantEmbedding into high-level semantic representations, which can be formulated as:

$$\mathbf{y}_i^{\text{topology}} = \text{MLP}_t(\text{NetworkEmbedding}(\mathcal{G}, i)), \tag{2}$$

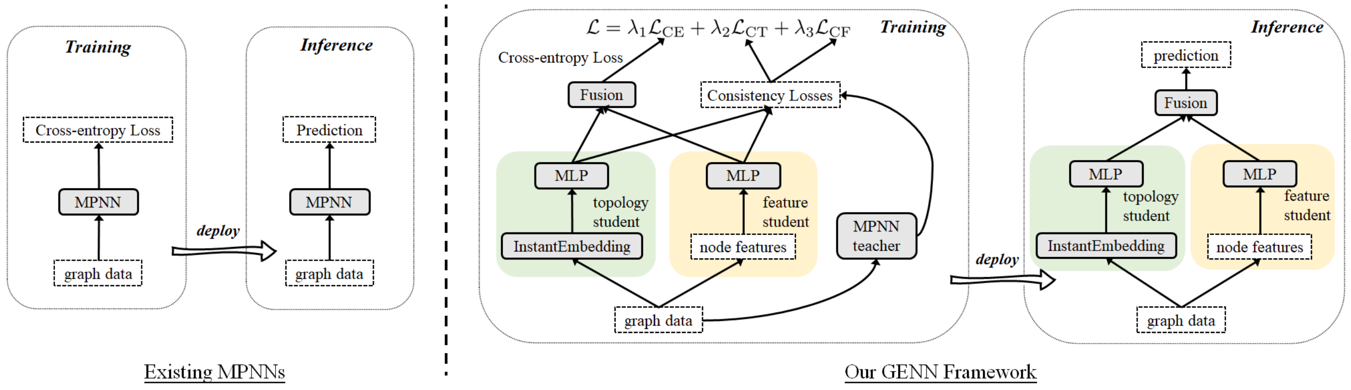


Figure 2: (*left*) The training and inference of existing MPNNs. (*right*) Our GENN framework. During training, we supervise the topology and feature students through distilled knowledge from the MPNN teacher. During inference, the MPNN teacher is not used. We deploy the distilled students for inference, which is more efficient and effective than the MPNN teacher.

where $\mathcal{G}$ is the input graph topology, and $\mathbf{y}_i^{\text{topology}} \in \mathbb{R}^C$ is the predicted logit vector taking the input of only the graph topology. MLP_t is the MLP for encoding graph topology, for which we set the number of layers as same as MPNN by default for a fair comparison.

Obtaining $\mathbf{y}_i^{\text{topology}}$ is fast and easy for deployment since it does not require the massive multiplication-and-accumulation (MAC) operations on neighbors' features like the popular MPNNs. The efficiency of our topology student benefits from the developments of NE methods [3, 51], e.g., InstantEmbedding [33]. There exist other NE options for the implementation [3, 51], but we use InstantEmbedding in our implementation, owing to its promising efficiency and scalability to large graphs.

Feature Student. We construct a multi-layer perceptron (MLP) module to encode the input node-wise features as high-level semantic representations, which can be formulated as:

$$\mathbf{y}_i^{\text{feature}} = \text{MLP}_f(\mathbf{x}_i), \tag{3}$$

where $\mathbf{x}_i$ is the input node feature node i, and $\mathbf{y}_{i,\text{feat}} \in \mathbb{R}^C$ is the predicted logit vector taking the input of only the node features, where C is the number of classes. MLP_f is the MLP for encoding node features, for which we set the number of layers as same as MPNN by default for a fair comparison. Obtaining $\mathbf{y}_i^{\text{feature}}$ is fast and amenable to industrial deployment, as it sidesteps problems induced by graph dependency [21, 50].

Fusion Head. Given the semantic embeddings retrieved from two student modules: $\mathbf{y}_i^{\text{topology}}, \mathbf{y}_i^{\text{feature}}$, GENN makes the final predictions by fusing them. Since the ground-truth labels can be more strongly correlated with either graph topology or features [31], we employ the attention mechanism [43] $\text{Attention}(\mathbf{y}_i^{\text{topology}}, \mathbf{y}_i^{\text{feature}})$ to learn the corresponding importance $\alpha_i^{\text{topology}}, \alpha_i^{\text{feature}}$ for graph topology and features as follows:

$$(\alpha_i^{\text{topology}}, \alpha_i^{\text{feature}}) = \text{Attention}(\mathbf{y}_i^{\text{topology}}, \mathbf{y}_i^{\text{feature}}), \tag{4}$$

where $\alpha_i^{\text{topology}}, \alpha_i^{\text{feature}}$ indicates the attention values of node i with embeddings $\mathbf{y}_i^{\text{topology}}, \mathbf{y}_i^{\text{feature}}$ respectively. In detail, we first transform the logits $\mathbf{y}_i^{\text{topology}}, \mathbf{y}_i^{\text{feature}}$ through a nonlinear transformation, and then use an attention vector $\mathbf{q}$ to get the attention weights $w_i^{\text{topology}}, w_i^{\text{feature}}$ as follows:

$$w_i^{\text{topology}} = \mathbf{q}^T \cdot \tanh(\mathbf{W} \cdot \mathbf{y}_i^{\text{topology}} + \mathbf{b}) \tag{5}$$

Here, $\mathbf{W}$ is the weight matrix and $\mathbf{b}$ is the bias vector. Similarly, we can get the attention values w_i^{feature} for node i on the embedding $\mathbf{y}_i^{\text{topology}}$ accordingly. We then normalize the attention values $w_i^{\text{topology}}, w_i^{\text{feature}}$ with the softmax function to get the final weight:

$$\alpha_i^{\text{topology}} = \text{softmax}(w_i^{\text{topology}}) = \frac{\exp(w_i^{\text{topology}})}{\exp(w_i^{\text{topology}}) + \exp(w_i^{\text{feature}})} \tag{6}$$

Larger $\alpha_i^{\text{topology}}$ implies greater importance of the corresponding module. Similarly, $\alpha_i^{\text{feature}} = \text{softmax}(w_i^{\text{feature}})$ holds. Then we attend different logits with the learned attention weights to obtain the final prediction:

$$\mathbf{y}_i = \alpha_i^{\text{topology}} \cdot \mathbf{y}_i^{\text{topology}} + \alpha_i^{\text{feature}} \cdot \mathbf{y}_i^{\text{feature}}. \tag{7}$$

This attention based fusion head is analogous to the classical attention modules [41, 43]. Some works use the MLP following a concatenation of the input embeddings as the fusion head [35]; this kind of MLP-based fusion has the fusion weights related to the channels instead of the specific input embeddings, which makes it difficult to assign adaptive weights for the embeddings of different nodes. The empirical results validate our choice as well: Table 8 shows that the Attention based fusion head is more effective than the MLP fusion on node classification.

Optimization Objective. Our optimization includes two parts: the supervised cross-entropy loss and two consistency losses for KD. The former is

$$\mathcal{L}_{\text{CE}} = \frac{1}{|\mathcal{V}_L|} \sum_{i \in \mathcal{V}_L} \mathcal{H}(\mathbf{y}_i^\star, \mathbf{y}_i), \tag{8}$$

where $\mathcal{H}(\cdot,\cdot)$ is the cross-entropy function [12], $\mathcal{V}_L$ is the set of labeled nodes, and $\mathbf{y}_i^\star$ is the ground truth label of node i. In addition to the supervised signals of $\mathcal{L}_{CE}$, we apply consistency losses to supervise the topology and feature students respectively using pseudo-labels given by the MPNN teacher $\mathbf{y}_i^{\text{teacher}}$:

$$\mathcal{L}_{\text{CT}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathcal{D}_{\text{KL}}(\mathbf{y}_i^{\text{teacher}} \| \mathbf{y}_i^{\text{topology}}), \tag{9}$$

$$\mathcal{L}_{\text{CF}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathcal{D}_{\text{KL}}(\mathbf{y}_i^{\text{teacher}} \| \mathbf{y}_i^{\text{feature}}) \tag{10}$$

where $\mathcal{D}_{\text{KL}}(\cdot\|\cdot)$ is the Kullback-Leibler divergence [22] measuring the divergence between two distributions, and $\mathcal{V}$ is the set of nodes. By minimizing $\mathcal{L}_{\text{CT}}$ and $\mathcal{L}_{\text{CF}}$, we encourage both the topology and feature students to be as effective as the MPNN teacher.

Overall, combining the cross-entropy and the consistency losses, we have the objective function:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{CE}} + \lambda_2 \mathcal{L}_{\text{CT}} + \lambda_3 \mathcal{L}_{\text{CF}} \tag{11}$$

We adaptively set the values of λ_1, λ_2, and λ_3 for different datasets based on the grid beam search [18] in a scoped space $\lambda_1, \lambda_2, \lambda_3 \in [a, b]$ where a, b are the boundaries of the search range. We search the values once on the validation set of each dataset, and use the fixed values for inference on all testing instances.

Unlike most KD work which uses the final teacher prediction $\mathbf{y}$ to teach the student [13], we apply the supervision from the teacher MPNN to two student modules separately. The advantage of our design is two-fold: First, our design endows GENN with higher flexibility on supervision; we can set different weights to the specific student module to adaptively learn from the ground-truth labels or the teacher model. Second, our loss design gives our GENN better modularity. Every distilled student module can independently make effective predictions by itself. Besides, the effectiveness of each student module forms the basis of highly accurate final predictions.

Inference. As analyzed in Sec. 1, MPNNs hold high inference latency due to the implicit use of graph topology. Thus, during inference, we do not use MPNNs but only uses the fused output $\mathbf{y}_i$ from our distilled student modules as the prediction, as shown in Fig. 2 right. This leads to much more efficient inference than the original MPNNs. Our GENN's inference only includes the fast computations on InstantEmbedding, two MLPs, and a lightweight attention head, which saves the massive MAC operations of MPNNs on many neighbors' features and thus significantly reduces the inference latency (see Fig. 1).

Discussion. Our GENN framework can be seen as leveraging the best parts of current NE methods, MPNNs, and MLPs to achieve strong practical desiderata which neither one can independently achieve. Notice that the MPNNs are the state-of-the-art models on graph learning given rich features thanks to their complex architectures. Conversely, they struggle in feature- and latency- constrained settings. We compensate for the first weakness by explicitly encoding graph topology using InstantEmbedding and an MLP. We next compensate for the second weakness by reducing MAC operations over neighbors' features during inference while simultaneously leveraging the inductive bias advantages of MPNNs with KD. Despite the seemingly straightforward technical choices, the careful utility of these components together achieves a very practically strong result while greatly alleviates deployability bottlenecks. In summary, our GENN is designed to be highly accurate, inductive, quick during inference, and able to outperform its MPNN teachers in both feature-constrained and feature-rich settings via its explicit encoding with adaptive fusion.

Table 1: Dataset statistics.

Dataset	#Nodes	#Edges	#Classes	#Attributes
Cora	2,708	5,429	7	1,433
Citeseer	3,327	4,732	6	3,703
Pubmed	19,717	44,338	3	500
CoraML	2,995	16,316	7	2,879
OGB-Arxiv	169,343	1,166,243	40	128
OGB-Products	2,449,029	61,859,140	47	100

4 EXPERIMENTS

In this section, we evaluate the performance of our GENN framework. We compare our method with MPNN teachers on the task of node classification. We report experimental results under both the transductive and inductive settings [50]. Then, we adjust the ratio of the accessible node features to evaluate the generalization of graph learning models under limited node features. In addition, we visualize the distribution of attention weights to analyze whether GENN can learn appropriate attention weights on different datasets. Besides, we compare the inference efficiency of our GENN and other acceleration methods on the MPNN models. Last but not least, we conduct the ablation studies to analyze the effects of optimization targets and the fusion head. Our experimental settings closely follow those of the previous work [20, 46, 50] to ensure a fair comparison.

We use the standard benchmark datasets Cora, Citeseer, Pubmed [47], CoraML [1], OGB-Arxiv, and OGB-Products [20] for evaluation. The first four are citation networks, where each node is a document and each edge is a citation link. OGB-Arxiv is extracted from the Microsoft Academic Graph (MAG) [42], which is a paper citation network of arXiv papers. OGB-Products is an Amazon products co-purchasing network originally developed by [5]. The statistics of these datasets are summarized in Table 1. For the datasets where the standard split is available [20, 47], we follow the standard split for a fair comparison. Otherwise, we randomly split the nodes in the whole graph as 30%, 20%, 50% for training, validation, and testing respectively [1].

For all experiments in this section, we report the average and standard deviation over ten runs with different random seeds. Model performance is measured as accuracy, and results are reported on the the best model selected using validation data. We set hyperparameters of the used techniques and considered baseline methods, e.g., the batch size, the number of hidden units, the optimizer, and the learning rate as suggested by their authors. For the hyperparameters of our GENN method, we set the search range of the hyperparameters in Eq. (11) as $[0, 2]$ and the search step 0.1. We run all experiments on a machine with 80 Intel(R) Xeon(R) E5-2698 v4 @ 2.20GHz CPUs, and a single NVIDIA V100 GPU with 16GB RAM.

4.1 Performance with the SAGE Teacher

For a fair comparison, we use the popular MPNN model GraphSAGE (short as SAGE) [16] as the teacher model, to evaluate whether

Table 2: Test accuracy of the transductive node classification of GENN under the supervision of the teacher MPNN SAGE and the baseline methods. We conduct 10 trials with random weight initialization. The mean and standard deviations are reported. The best results in each row are highlighted in bold font. 's/' denotes 'supervised by'. Δ denotes the improvements of our GENN to the corresponding baselines.

Dataset	InstantEmbedding [33]	MLP [36]	SAGE [16]	GLNN [50] s/ SAGE	GENN (Ours) s/ SAGE	Δ SAGE	Δ InstantEmbedding
Cora	70.6 ± 0.9	55.1 ± 0.6	81.5 ± 0.7	80.5 ± 1.1	**84.0 ± 0.8**	↑ 3.1%	↑ 19.0%
Citeseer	49.4 ± 1.2	46.5 ± 0.9	70.3 ± 0.7	71.4 ± 0.9	**73.1 ± 0.8**	↑ 4.0%	↑ 48.0%
Pubmed	69.1 ± 1.0	71.4 ± 0.8	79.0 ± 0.8	75.4 ± 1.2	**80.3 ± 0.9**	↑ 1.6%	↑ 16.2%
CoraML	68.2 ± 1.2	36.4 ± 1.2	48.9 ± 1.0	47.2 ± 1.4	**74.3 ± 1.1**	↑ 51.9%	↑ 8.9%
OGB-Arxiv	65.9 ± 0.3	55.5 ± 0.4	70.9 ± 0.2	63.5 ± 0.5	**72.1 ± 0.4**	↑ 1.7%	↑ 9.4%
OGB-Products	65.1 ± 0.5	61.1 ± 0.1	78.6 ± 0.5	68.9 ± 0.5	**79.2 ± 0.4**	↑ 0.8%	↑ 21.7%

Table 3: Test accuracy of the inductive node classification of GENN under the supervision of the teacher MPNN SAGE and the baseline methods. We conduct 10 trials with random weight initialization. The mean and standard deviations are reported. The best results in each row are highlighted in bold font. 's/' denotes 'supervised by'. Δ denotes the improvements of our GENN to the corresponding baselines.

Dataset	InstantEmbedding [33]	MLP [36]	SAGE [16]	GLNN [50] s/ SAGE	GENN (Ours) s/ SAGE	Δ SAGE	Δ InstantEmbedding
Cora	69.8 ± 0.8	55.1 ± 0.6	79.4 ± 0.8	73.8 ± 1.2	**81.5 ± 0.6**	↑ 2.6%	↑ 16.8%
Citeseer	49.0 ± 1.3	46.5 ± 0.8	69.7 ± 0.7	69.3 ± 0.8	**72.3 ± 0.8**	↑ 3.7%	↑ 47.6%
Pubmed	68.3 ± 0.8	71.4 ± 0.8	78.2 ± 0.9	74.3 ± 1.3	**80.1 ± 0.9**	↑ 2.4%	↑ 17.3%
CoraML	67.6 ± 1.0	36.4 ± 1.2	47.4 ± 1.3	45.6 ± 1.2	**71.9 ± 0.7**	↑ 51.7%	↑ 6.4%
OGB-Arxiv	65.3 ± 0.4	55.5 ± 0.4	70.6 ± 0.6	60.5 ± 0.5	**71.1 ± 0.5**	↑ 0.7%	↑ 9.0%
OGB-Products	64.8 ± 0.3	61.1 ± 0.1	76.5 ± 0.5	68.2 ± 0.4	**77.3 ± 0.3**	↑ 1.0%	↑ 19.3%

GENN can match or improve upon its teacher during inference. In addition, we also take the InstantEmbedding [33] and MLP [36] into comparison to evaluate whether we can leverage the best parts of InstantEmbedding, MLP, and SAGE to achieve strong practical performance and implications which none of them can independently achieve. We also compare the performance of GENN with the GLNN [50] model, which is a recently proposed KD method for graph learning.

We report the transductive node classification results in Table 2. We observe that our GENN, which employs SAGE as the teacher model, outperforms the teacher SAGE by 3.1% on Cora, 4.0% on Citeseer, 1.6% on Pubmed, 51.9% on CoraML, 1.7% on OGB-Arxiv, 0.8% on OGB-Products. As a result, our GENN outperforms all the baseline methods including its teacher SAGE and the advanced knowledge distillation method GLNN on the transductive node classification. Additionally, we report the inductive node classification performance in Table 3. The similar improvements achieved by our GENN are observed. The above experimental results validate the effectiveness of our proposed GENN on learning from the MPNN teacher of SAGE and improve the MPNN teacher for more accurate inference based on our explict encoding and adaptive fusion [11].

Among other baseline methods, MLP only utilizes node features for graph learning while the network embedding method InstantEmbedding only utilizes the graph topology. Both do not fully utilize the information in the input graph and thus cannot achieve the best performance. GLNN also utilizes the SAGE model as the teacher for supervision. However, it does not utilize the graph topology for inference and thus is not as effective as our GENN. Overall, our GENN model explicitly encodes the graph topology and features, and adaptively fuses these two main sources of information for graph learning, which outperforms the MPNN teacher and achieves superior effectiveness over the baseline methods.

4.2 Performance Given Limited Node Features

In the practical applications, graph learning models usually have limited access to node features, either for reasons of privacy, fairness or availability [27, 29, 39]. For example, in a social network application, the model may not be able to use certain users' sensitive features (e.g. underage users) [6], or use data from users in very sparsely populated cities (e.g. to satisfy k-anonymity) [39]. Nowadays, these cases are increasing because of the higher privacy protection demands of most people and more strict regulations on the platforms [30, 30]. In this case, the graph learning models are desired to exploit the graph topology better for more effective predictions although the node features are limited.

To evaluate whether the graph learning models can generalize well to particularly challenging cases where node features are limited, we propose a filtered evaluation setting, where we retain partial node features for evaluation in the transduction node classification. In this setting, the graph learning models cannot overly rely on the node features for node classification but have to utilize the graph topology effectively for node classification. In our experiments, we randomly remove the node features in terms of the input channels with specific ratios and present the evaluation results on the limited node features in Table 4. Our GENN method is much more effective than its teacher SAGE given the limited node features. This setting is as same as that corresponding to Fig.

Table 4: Test accuracy under different ratio of accessible node features. We conduct 10 trials with random weight initialization and report the mean results. ↑ denotes performance improvements over the SAGE teacher.

Dataset	Ratio of Features	SAGE [16]	GENN (Ours) s/ SAGE
Cora	2%	35.2	72.5 (**↑106%**)
	5%	40.7	72.9 (**↑79%**)
	10%	46.3	73.5 (**↑59%**)
Citeseer	2%	36.5	51.4 (**↑41%**)
	5%	43.3	52.1 (**↑20%**)
	10%	47.2	52.3 (**↑11%**)
Pubmed	2%	51.3	72.6 (**↑41%**)
	5%	58.7	73.2 (**↑25%**)
	10%	64.0	74.0 (**↑16%**)
CoraML	2%	26.1	72.8 (**↑179%**)
	5%	27.5	73.6 (**↑168%**)
	10%	28.3	74.1 (**↑162%**)
OGB-Arxiv	2%	43.2	68.3 (**↑58%**)
	5%	60.8	68.8 (**↑13%**)
	10%	65.5	69.1 (**↑5%**)
OGB-Products	2%	22.3	68.0 (**↑205%**)
	5%	36.5	68.2 (**↑87%**)
	10%	46.9	68.5 (**↑46%**)

Table 5: Test accuracy of transductive node classification without the input node features (only the identity matrix as the node features). We conduct 10 trials with random weight initialization and report the mean results. ↑ denotes the performance improvements.

Dataset	SAGE [16]	GENN (Ours) s/ SAGE
Cora	63.9	72.1 (**↑13%**)
Citeseer	34.4	51.0 (**↑48%**)
Pubmed	49.2	70.0 (**↑42%**)
CoraML	44.8	66.2 (**↑48%**)
OBG-Arxiv	59.5	68.1 (**↑14%**)
OBG-Products	51.9	67.9 (**↑31%**)

1 left. Under this challenging setting, the improvements achieved by our GENN are much higher than those with the rich features as presented in Table 2.

Specifically, when the ratio of accessible node features is 2%, GENN outperforms its teacher SAGE by more than 40% on all the datasets, and even achieve the 179% and 205% improvements on CoraML and OGB-Products respectively. Some prior work also suggests to use the one-hot vectors as the input node features when the input node features are limited [24, 48, 49]. We follow this setting to replace the original node features with the one-hot vectors. We report the experimental results in this setting in Table 5. Our GENN is still significantly more effective than its teacher SAGE given the one-hot input node features.

When the node features are limited, graph learning models have to utilize more informative information from graph topology to produce effective predictions. The MPNNs only implicitly uses the graph topology as the "pathways" to transmit the input node features, which cannot extract the valuable information from graph topology well given limited node features. In contrast, our GENN explicitly utilizes the graph topology as an independent source of information, which supports the model to perform well although the node feature are not rich. Overall, these experimental results validates the effectiveness of our method to learn from graph topology and the generalization of our method to the limited features.

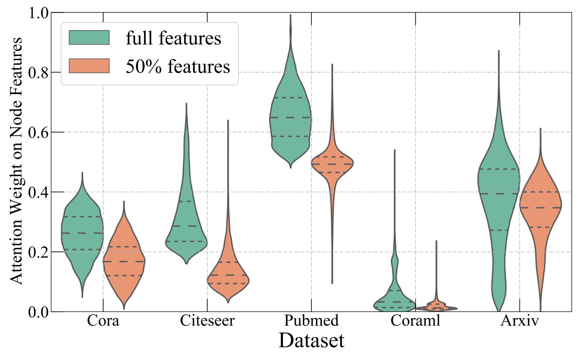


Figure 3: Analysis of attention distribution. We visualize the attention weights on the feature student in the y-axis through the violin plots. When the node features are limited, our attention based fusion adaptively assign higher weights to the graph topology, which is relatively more informative.

4.3 Analaysis of The Attention Fusion

In order to investigate whether the attention values learned by our proposed model are meaningful, we analyze the attention distribution. GENN learns two specific embeddings for graph topology and features respectively, each of which is associated with the attention values. We conduct the attention distribution analysis on different datasets in the transductive node classification experiment, of which the results are shown in Fig. 3. As we can see, for the Cora, Citeseer, CoraML, and OGB-Arxiv datasets, the attention values of the embeddings for the node features are smaller than 0.5, and thus lower than the weights for graph topology. This implies that the information in graph topology is more important than that in the node features. To verify this, we can see that the results of MLP is worse than InstantEmbedding on these datasets in Table 2 and 3. The former only uses the node features for prediction while the latter only uses the graph topology. Conversely, for Pubmed, in comparison with Table 2, 3, and Fig. 3, we find that MLP performs better than InstantEmbedding. Meanwhile, the attention values of the node embeddings on the node features are also larger than 0.5, i.e., larger than those for graph topology.

When we remove 50% node features, the attention weights on the embeddings for node features consistently decrease in all datasets. The reason is that when node features are limited, they provide less valuable information for predictions. Our attention module can adapt the weights to this change and learn the adaptive weights accordingly. In summary, these experimental results demonstrate that our proposed GENN can adaptively assign larger weights to graph topology or node features when deemed more important.

Table 6: While other inference acceleration methods speed up SAGE, they are considerably slower than our GENN. We report inductive inference time (in ms) on 10 randomly chosen nodes. ↑ denotes the times of acceleration over SAGE.

Method	OGB-Arxiv		OGB-Products	
	Acc. (%)	Time (ms)	Acc. (%)	Time (ms)
APPNP [25]	70.5	471.2	75.4	1989.4
SGC [45]	69.8	432.8	75.1	1902.3
SAGE [16]	70.6	493.6	76.5	2092.1
QSAGE [52]	70.1	441.2 (1.12×)	75.6	1973.8 (1.06×)
PSAGE [53]	70.3	457.1 (1.08×)	76.4	2011.9 (1.04×)
GENN (Ours) s/ SAGE	**71.1**	**12.5 (39.5 ×)**	**77.3**	**26.7 (78.4×)**

4.4 Inference Efficiency Evaluation

Common techniques of inference acceleration for GNNs include pruning [17, 53], quantization [15, 52], and model architecture simplifying, e.g., SGC, APPNP. These approaches can reduce model parameters and Multiplication-and ACcumulation (MACs) operations. But they do not eliminate the neighbor-fetching latency of message passing. We show an inductive inference speed comparison between SGC [45], APPNP [25], SAGE, quantized SAGE from FP32 to INT8 (QSAGE) [52], SAGE with 50% weights pruned (PSAGE) [53], and our GENN supervised by SAGE in Table 6.

With the same setting as Fig. 1 right, we observe that our GENN is considerably faster than the baseline methods, which achieves 40× - 78× acceleration for the SAGE teacher. This result demonstrates the advantage of our GENN on improving the inference efficiency by eliminating the massive MAC operations over the features of many neighbors from MP. Meanwhile, our GENN outperforms the teacher SAGE and other accelration techniques in terms of the effectiveness, which validates of the benefits from our explicit encoding of graph topology and the distilled knowledge from the teacher SAGE on guiding our GENN students to make effective predictions.

4.5 Ablation Study

We conduct ablation studies to empirically examine the effects of the optimization targets and the fusion head.

We report the experimental results of the ablation study on the optimization targets, i.e., the cross entropy loss $\mathcal{L}_{\text{CE}}$ and the consistency losses $\mathcal{L}_{\text{CT}}$, $\mathcal{L}_{\text{CF}}$ on the graph topology and features, in Table 7. We observe that the consistency losses on the graph topology influence more on the performance when the node features are complete. This validates the importance of effectively utilizing the graph topology for graph learning. Minimizing our consistency loss encourages the node representations on graph topology to be consistent to the teacher model's outputs, which produces rich supervision for our model. Moreover, we observe that removing the supervised cross-entropy loss causes serious performance degradation when the node features are limited. The key reason is that when the node features are limited, the teacher model no longer provides informative supervision signals. In this case, GENN has to learn from the ground-truth labels to produce effective predictions.

Table 8 presents the performance of GENN with the Attention Fusion and the alternate MLP fusion [35], which is a MLP following the concatenation of the two students outputs. The Attention

Table 7: We analyze the effects of the cross-entropy loss $\mathcal{L}_{\text{CE}}$, the consistency losses on graph topology: $\mathcal{L}_{\text{CT}}$, the loss on node features $\mathcal{L}_{\text{CF}}$, and the KD (including both $\mathcal{L}_{\text{CT}}$ and $\mathcal{L}_{\text{CF}}$), on the OGB-Arxiv dataset. 'w/o' denotes 'without'.

Method	Full Features	10% Features
GENN	**72.1 ± 0.4**	**69.1 ± 0.5**
w/o $\mathcal{L}_{CE}$	71.4 ± 0.5	65.8 ± 0.4
w/o $\mathcal{L}_{\text{CT}}$	70.7 ± 0.3	69.0 ± 0.5
w/o $\mathcal{L}_{\text{CF}}$	71.2 ± 0.4	68.9 ± 0.4
w/o KD	70.3 ± 0.1	68.7 ± 0.4

Table 8: We compare the effects of the fusion head as our Attention based one and a MLP alternate [35].

Fusion	Pubmed	CoraML	OGB-Products
MLP [35]	77.3 ± 1.0	68.7 ± 1.2	71.3 ± 0.5
Attention (Ours)	**80.3 ± 0.9**	**74.3 ± 1.1**	**79.2 ± 0.4**

Fusion outperforms the MLP fushion for the transductive node classification on differrent datasets. The reason is that the MLP based fusion has the fusion weights related to the channels instead of the specific input embeddings, which makes it difficult to assign adaptive weights for the embeddings of different nodes. In contrast, the Attention fusion adaptively learns the fusion weights corresponding to the specific students' outputs, which adapts our GENN to flexibly attend to the more important information between the graph topology and features for different instances.

5 CONCLUSION

In this paper, we propose a graph learning framework, GENN, which improves over state-of-the-art MPNNs to achieve accurate and efficient predictions at inference. GENN addresses two key issues of MPNNs which hinders their large-scale industrial applications: over-reliance on node features and high inference latency. Specifically, GENN explicitly encodes the graph topology as an important source of information, which mitigates the over-reliance on node features and significantly improve the accuracy of MPNNs by more than 40% when features are limited. Besides, GENN eliminates the massive fetching and transformation over many neighbors' features, which improves the inference efficiency of MPNNs by 40×-78×. Last but not least, thanks to the distilled knowledge applied to our GENN students, GENN achieves better effectiveness than the teacher MPNNs even when the input features are rich. We conduct a comprehensive study of GENN's empirical properties; promising results on multiple graph benchmarks show that GENN is a handy choice for deployment of graph learning approaches in the presence of feature or latency constraints.

ACKNOWLEDGMENTS

This paper is supported by NUS ODPRT Grant R252-000-A81-133 and Singapore Ministry of Education Academic Research Fund Tier 3 under MOEs official grant number MOE2017-T3-1-007.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[3] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. 2019. Fast and accurate network embeddings via very sparse random projection. In *Proceedings of the 28th ACM international conference on information and knowledge management.* 399–408.

[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning.* PMLR, 1725–1735.

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM KDD.* 257–266.

[6] Hui-Lien Chou, Yih-Lan Liu, and Chien Chou. 2019. Privacy behavior profiles of underage Facebook users. *Computers & Education* 128 (2019), 473–485.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *NeurIPS* 29 (2016).

[8] Chi Thang Duong, Thanh Dat Hoang, Ha The Hien Dang, Quoc Viet Hung Nguyen, and Karl Aberer. 2019. On node features for graph neural networks. *arXiv preprint arXiv:1911.08795* (2019).

[9] Yoav Einav. 2019. Amazon Found Every 100ms of Latency Cost them 1% in Sales (https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales). (2019).

[10] Lukas Faber, Yifan Lu, and Roger Wattenhofer. 2021. Should Graph Neural Networks Use Features, Edges, Or Both? *arXiv preprint:2103.06857* (2021).

[11] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *International Conference on Machine Learning.* PMLR, 1607–1616.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning.* MIT press.

[13] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.

[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM KDD.* 855–864.

[15] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International conference on machine learning.* PMLR, 1737–1746.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS* 30 (2017).

[17] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *NeurIPS* 28 (2015).

[18] Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *arXiv preprint arXiv:1704.07138* (2017).

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS* 33 (2020), 22118–22133.

[21] Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. 2021. Graph-MLP: node classification without message passing in graph. *arXiv preprint arXiv:2106.04051* (2021).

[22] James M Joyce. 2011. Kullback-leibler divergence. In *International encyclopedia of statistical science.* Springer, 720–722.

[23] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M Aamodt, Natalie Enright Jerger, and Andreas Moshovos. 2016. Proteus: Exploiting numerical precision variability in deep neural networks. In *Proceedings of the 2016 International Conference on Supercomputing.* 1–12.

[24] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[25] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[26] Konrad Kollnig, Anastasia Shuba, Max Van Kleek, Reuben Binns, and Nigel Shadbolt. 2022. Goodbye tracking? Impact of iOS app tracking transparency and privacy labels. *arXiv preprint arXiv:2204.03556* (2022).

[27] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. 2006. Mondrian multidimensional k-anonymity. In *22nd International conference on data engineering (ICDE'06).* IEEE, 25–25.

[28] Yawei Li, He Chen, Zhaopeng Cui, Radu Timofte, Marc Pollefeys, Gregory S Chirikjian, and Luc Van Gool. 2021. Towards efficient graph convolutional networks for point cloud handling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 3752–3762.

[29] Sonia Livingstone, Kjartan Ólafsson, and Elisabeth Staksrud. 2013. Risky social networking practices among "underage" users: Lessons for evidence-based policy. *Journal of Computer-Mediated Communication* 18, 3 (2013), 303–320.

[30] Alba Ribera Martínez. 2022. Trading Off the Orchard for an Apple: the iOS 14.5 Privacy Update. *Journal of European Competition Law & Practice* 13, 3 (2022), 200–216.

[31] John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. 2022. GraphWorld: Fake Graphs Bring Real Insights for GNNs. *arXiv preprint arXiv:2203.00112* (2022).

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM KDD.* 701–710.

[33] Ştefan Postăvaru, Anton Tsitsulin, Filipe Miguel Gonçalves de Almeida, Yingtao Tian, Silvio Lattanzi, and Bryan Perozzi. 2020. InstantEmbedding: Efficient local node representations. *arXiv preprint arXiv:2010.06992* (2020).

[34] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining.* 459–467.

[35] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM conference on recommender systems.* 240–248.

[36] Martin Riedmiller and AM Lernen. 2014. Multi layer perceptron. *Machine Learning Lab Special Lecture, University of Freiburg* (2014), 7–24.

[37] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.

[38] Gian Luca Scoccia, Marco Autili, Giovanni Stilo, and Paola Inverardi. 2022. An empirical study of privacy labels on the Apple iOS mobile app store. (2022).

[39] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[40] Joshua B Tenenbaum, Vin de Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[42] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[43] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM KDD.* 1243–1253.

[44] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM KDD.* 207–217.

[45] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning.* PMLR, 6861–6871.

[46] Cheng Yang, Jiawei Liu, and Chuan Shi. 2021. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the Web Conference 2021.* 1227–1237.

[47] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning.* PMLR, 40–48.

[48] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. 2021. Identity-aware graph neural networks. *arXiv preprint arXiv:2101.10320* (2021).

[49] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *International Conference on Machine Learning.* PMLR, 7134–7143.

[50] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2022. Graph-less neural networks: Teaching old mlps new tricks via distillation. *ICLR* (2022).

[51] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-scale network embedding with iterative random projection. In *2018 IEEE International Conference on Data Mining (ICDM).* IEEE, 787–796.

[52] Yiren Zhao, Duo Wang, Daniel Bates, Robert Mullins, Mateja Jamnik, and Pietro Lio. 2020. Learned low precision graph neural networks. *arXiv preprint arXiv:2009.09232* (2020).

[53] Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. 2021. Accelerating large scale real-time GNN inference using channel pruning. *arXiv preprint arXiv:2105.04528* (2021).

[54] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. *NeurIPS* 32 (2019).