Meta-Aggregator: Learning to Aggregate for 1-bit Graph Neural Networks

Yongcheng Jing[1], Yiding Yang[2], Xinchao Wang[3], Mingli Song[4], Dacheng Tao[5,1]
[1]The University of Sydney, Australia, [2]Stevens Institute of Technology,
[3]National University of Singapore, [4]Zhejiang University, [5]JD Explore Academy, China
yjin9495@sydney.edu.au, yyang99@stevens.edu,
xinchao@nus.edu.sg, brooksong@zju.edu.cn, dacheng.tao@gmail.com

Abstract

In this paper, we study a novel meta aggregation scheme towards binarizing graph neural networks (GNNs). We begin by developing a vanilla 1-bit GNN framework that binarizes both the GNN parameters and the graph features. Despite the lightweight architecture, we observed that this vanilla framework suffered from insufficient discriminative power in distinguishing graph topologies, leading to a dramatic drop in performance. This discovery motivates us to devise meta aggregators to improve the expressive power of vanilla binarized GNNs, of which the aggregation schemes can be adaptively changed in a learnable manner based on the binarized features. Towards this end, we propose two dedicated forms of meta neighborhood aggregators, an exclusive meta aggregator termed as Greedy Gumbel Neighborhood Aggregator (GNA), and a diffused meta aggregator termed as Adaptable Hybrid Neighborhood Aggregator (ANA). GNA learns to exclusively pick one single optimal aggregator from a pool of candidates, while ANA learns a hybrid aggregation behavior to simultaneously retain the benefits of several individual aggregators. Furthermore, the proposed meta aggregators may readily serve as a generic plugin module into existing full-precision GNNs. Experiments across various domains demonstrate that the proposed method yields results superior to the state of the art.

1. Introduction

Graph neural networks (GNNs) have recently emerged as the dominant paradigm for learning and analyzing non-Euclidean data, which contain rich node content information as well as topological relational information [7, 12, 53]. As such, a massive number of GNN architectures have been developed [21, 46, 55, 60, 63]. The success of GNNs also triggers a great surge of interest in applying elaborated graph networks to various tasks across many domains, such as object detection [11, 8], pose estimation [59], point cloud processing [22, 51, 36], and visual SLAM [40]. These

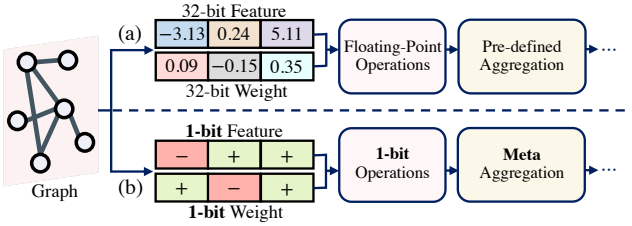

Figure 1. Illustrations of the computational workflow in (a) conventional full-precision GNNs and (b) the proposed 1-bit GNNs. In particular, we devise two meta aggregators for the proposed model, termed as *Greedy Gumbel Aggregator* (GNA) and *Adaptable Hybrid Aggregator* (ANA), that learn to perform adaptive aggregation in a graph-aware and layer-aware manner.

GNN-based applications, in general, rely on cumbersome graph architectures to deliver gratifying results. For example, SuperGlue, a GNN-based feature matching approach, requires 12M network parameters to achieve the state-of-the-art performance [40].

In practice, however, such applications typically require a compact and lightweight architecture for real-time interaction, especially in resource-constrained environments. In the case of autonomous driving [30], for example, it is critical to maintain fast and timely responses for GNN-based SLAM algorithms to handle complex traffic conditions, thereby leading to the urgent need of compressing cumbersome GNN models. The work of [58], as the first attempt, leverages knowledge distillation to learn a compact student GNN with fewer parameters. In spite of the improved efficiency, this approach still relies on the expensive floating-point operations, let alone a well-performed teacher model pre-trained in the first place.

In this paper, we strive to make one step further towards ultra lightweight GNNs. Our goal is to train a customized 1-bit GNN, as shown in Fig. 1, that allows for favorable memory efficiency and meanwhile enjoys competitive performance. We start with developing a naïve GNN binarization framework, achieved through converting 32-bit features and parameters into 1-bit ones, followed by leveraging straight-

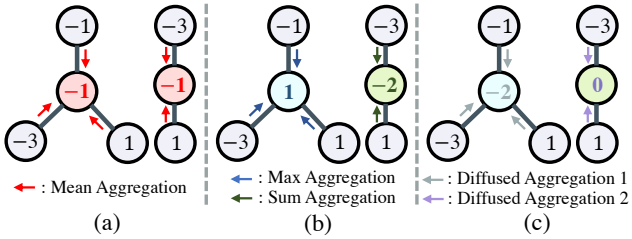

Figure 2. Example aggregation results of the two graphs with different topological structures for (a) the conventional pre-defined and fixed aggregator, (b) the proposed exclusive form of meta aggregators GNA, and (c) the proposed diffused form of meta aggregators ANA.

through estimator to optimize the binarized model. The derived vanilla binarized GNN enjoys favorable memory efficiency; however, its performance is not encouraging as expected. Through parsing its underlying process, we identified that the binarization yields limited expressive power, making the model incapable to distinguish different graph topologies. An illustrating example is shown in Fig. 2(a), where a *mean* aggregator, which is commonly adopted by full-precision GNNs, produce identical aggregation results for two diversified graph topologies with binarized features, thereby leading to inferior performances.

Inspired by this discovery, we introduce to the proposed GNN binarization framework a learnable and adaptive neighborhood aggregator, so as to alleviate the aforementioned dilemma and enhance the distinguishability of 1-bit graphs. Unlike existing GNNs that rely on a pre-defined and fixed aggregator, our elaborate meta neighborhood aggregators enables dynamically *selecting* (Fig. 2(b)) or *generating* (Fig. 2(c)) customized input- and layer-specific aggregation schemes. As such, we explicitly account for the customized characteristics of binarized graph features, and further strengthen the discriminative power for handling topological structures.

Towards this end, we propose two variants of meta aggregators: an exclusive meta aggregator, termed as *Greedy Gumbel Neighborhood Aggregator* (GNA), that adaptively *selects* an optimal aggregator in a learnable manner, as well as a diffused meta aggregator, termed as *Adaptable Hybrid Neighborhood Aggregator* (ANA), that either *approximates* a single aggregator or dynamically *generates* a hybrid aggregation behavior. Specifically, GNA incorporates the discrete decisions from the candidate aggregators, conditioned on the individual graph features, into the gradient descent process by leveraging *Greedy Gumbel Sampling*. Inevitably, the performance of GNA is bottlenecked by the individual aggregators in the candidate pool. Thus, we further devise ANA that enables generating a hybrid aggregator dynamically based on the input 1-bit graphs. ANA simultaneously preserves the strengths of multiple individual aggregators, leading to favorable competence to handle the challenging 1-bit graph features. Moreover, the proposed GNA and ANA can be readily extended as portable modules into the general full-precision GNN models to enhance the expressive capability.

In sum, our contribution is a novel GNN-customized binarization framework that generates a 1-bit lightweight GNN model with competitive performance, making it competent for resource-constrained applications such as edge computing. This is specifically achieved through an adaptive meta aggregation scheme to accommodate the challenging quantized graph features. We evaluate the proposed customized framework on several large-scale benchmarks across different domains and graph tasks. Experimental results demonstrate that the proposed meta aggregators achieve results superior to the state-of-the-art, not only on the devised 1-bit binarized GNN models, but also on the general full-precision models.

2. Related Work

We briefly review here several topics that are related to our work, including graph neural networks (GNNs), GNN-based applications as well as prior CNN-based network binarization techniques.

Graph Neural Networks. The concept of graph neural networks was proposed in [41], which generalized existing neural networks to handle graph data represented in the non-Euclidean domain. Over the past few years, graph neural networks have achieved unprecedented advances with various approaches being developed [21, 19, 63, 7, 57, 56, 25, 27, 48, 13, 24, 32, 26]. For example, graph attention network in [46] introduces a novel attention mechanism for efficient graph processing. GraphSAGE [10], on the other hand, addresses the scalability issues on large-scale graphs by sampling and aggregating feature representations from local neighborhoods.

The success of GNNs has also boosted the applications of graph networks in a wide range of problem domains [63], including semantic segmentation [52, 22, 36, 34], object detection [11, 8], pose estimation [59], interaction detection [35, 17], and visual SLAM [40], *etc*. Specifically, Wang *et al*. [52] propose a dynamic graph convolutional model for point cloud classification and semantic segmentation, which combines the advantages of the PointNet [33] and graph convolutional network [21]. Despite the encouraging performance, there is a lack of research on compressing cumbersome GNN models, which is critical for deployment in resource-constrained environments like on the mobile-terminal side.

Network Binarization. In the field of model compression [62, 42, 43, 5, 44], network binarization techniques aim to save memory occupancy and accelerate the network

inference by binarizing network parameters and then utilizing bitwise operations [14, 15, 4]. In recent years, various CNN binarization methods have been proposed, which can be categorized into direct binarization [6, 14, 15, 20] and optimization-based binarization [38, 4, 29]. Specifically, direct binarization quantizes the weights and activations to 1 bit with a pre-defined binarization function. In contrast, optimization-based binarization introduces scaling factors for the binarized parameters to improve the representation ability, but inevitably leading to inferior efficiency.

Driven by the success of the aforementioned binarization techniques in the CNN domain, in the paper, we propose a GNN-specific binarization method. Specifically, we primarily focus on GNN-based direct binarization, since our goal is to develop super lightweight GNN models. We also notice three concurrent works [47, 49, 1] that also aim to accelerate the forward process for GNN models. However, [47, 49] directly apply CNN-based binarization techniques without considering the characteristics of GNNs, which in fact will serve as the baseline method in our experiments. The other work in [1] only focuses on improving the efficiency of dynamic graph convolutional model [52], by speeding up the dynamic construction of k-nearest-neighbor graphs in the Hamming space. Unlike [47, 49, 1], we aim to devise a more general GNN-specific binarization framework that is applicable to most existing GNN models.

3. Vanilla Binary GNN and Pre-analysis

In this section, we first develop a vanilla binary GNN framework by simply binarizing model parameters and activations. We then show the limitations of this vanilla binary GNN by looking into the internal message aggregation process and accordingly develop two possible solutions to address these limitations. Eventually, built upon the possible solutions, we introduce the idea of the proposed customized GNN binarization framework with the meta aggregators.

Formulation of GNN Models. GNNs leverage graph topologies and node/edge features to learn a representation vector of a node, an edge or the entire graph. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a directed/undirected graph with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$, where $\{v_j\}$ is the set of neighboring nodes of v_i. Each node has an associated node feature $\mathcal{X} = [x_1\ x_2\ ...\ x_n]$. For example, in the task of 3D object classification, x can be set as the 3D coordinates.

Existing GNNs follow an iterative neighborhood aggregation scheme at each GNN layer, where each node v_i iteratively gathers features from its neighboring nodes $\{v_j\}$ to capture the structural information [23, 55]. Let $\mathcal{X}_i^\ell$ denote the feature vector of the node v_i at layer ℓ. The corresponding updated feature vector $\mathcal{X}_i^{\ell+1}$ in a GNN can then be formulated as:

$$\mathcal{X}_i^{\ell+1} = f\left(\mathcal{X}_i^\ell, \{\mathcal{X}_j^\ell : (j,i) \in \mathcal{E}\}\right), \tag{1}$$

where $\mathcal{X}_j^\ell$ represents the feature associated with the neighboring nodes. f is a mapping function that takes $\mathcal{X}_i^\ell$ as well as $\mathcal{X}_j^\ell$ as inputs. The choice of the mapping f corresponds to different architectures of GNNs.

For the sake of simplicity, we take here graph convolutional network (GCN) proposed by Kipf and Welling [21] as an example GNN architecture for the following discussions. We denote Mean as the mean aggregator that computes an average of the incoming messages and $\mathcal{W}$ as the learnable weight matrix for feature transformation. The general GNN form in Eq. 1 can then be instantiated for GCN as: $\mathcal{X}_i^{\ell+1} = \text{ReLU}\left(\mathcal{W}^l \text{Mean}_{(j,i)\in\mathcal{E}} \mathcal{X}_j^\ell\right)$ or $\mathcal{X}_i^{\ell+1} = \text{ReLU}\left(\text{Mean}_{(j,i)\in\mathcal{E}} \mathcal{W}^l \mathcal{X}_j^\ell\right)$, which respectively correspond to the case where aggregation comes first or comes after the feature transformation step [50].

Vanilla 1-bit GNN Models. We develop a naïve binarized GNN framework to compress cumbersome GNN models, by directly binarizing 32-bit input features and learnable weights in the feature transformation step into 1-bit ones.

Specifically, for the case of vanilla binary GCN, the forward propagation process can be modeled as:

$$\text{Net Forward:} \quad w_b = \text{sign}(w) = \begin{cases} +1, & w \geq 0 \\ -1, & w < 0 \end{cases}, \tag{2}$$

where w represents the element in the learnable weight matrix $\mathcal{W}$. We also binarize the graph features $\mathcal{X}$ in the same manner, by replacing w in Eq. 2 with the feature element x.

During the backward propagation, it is not feasible to simply exploit *Backward Propagation (BP)* algorithm [39], as most full-precision models do, to optimize binarized graph networks, due to the undifferentiable binarization function, *i.e.*, sign in Eq. 2. The derivative part of the sign function will lead to 0 gradients almost everywhere, thereby resulting in the vanishing gradient problem. To alleviate this dilemma, we leverage the *Straight-through Estimator (STE)* [2] for the backward propagation process in the binarized graph nets, formulated as:

$$\text{Net Backward:} \quad \frac{\partial \mathcal{L}}{\partial w} = \begin{cases} \frac{\partial \mathcal{L}}{\partial w_b}, & w \in (-1, 1) \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

where $\mathcal{L}$ represents the loss function. Essentially, Eq. 3 can be considered as propogagting the gradient through *hard tanh* function, defined as: $\text{Htanh}(x) = \text{Clip}(x, -1, 1)$.

We illustrate in Fig. 3 the computational workflow at an example binarized GCN layer for the case where the aggregation comes after the feature transformation. A similar scheme can be observed for the GCN model where the aggregation happens first. With compact node features and net weights, binarized GCN only relies on 1-bit XNOR and bit-count operations for graph-based processing, leading to an efficient and lightweight graph model that is competent for edge computing.

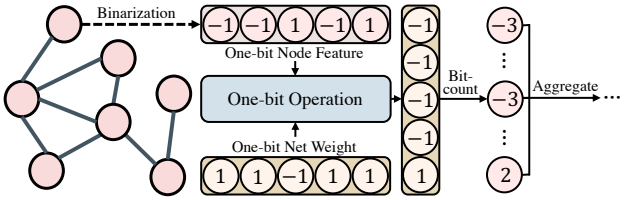

Figure 3. Illustrations of the computational workflow at an example binarized GNN layer. Despite the efficient 1-bit operations, the output features are less distinguishable between each other, leading to the challenge in the aggregation step shown in Fig. 4.

Challenges and Possible Solutions. Despite the compact binarized parameters and features, we empirically observed that the results of the developed vanilla GNN were not promising as expected. Specifically, we conduct a preliminary experiment on the ZINC dataset [16] with the GCN architecture in [7]. Averaged over 25 independent runs, the full-precision GCN model achieves the performance of 0.407 ± 0.018 in terms of the mean absolute error (MAE), whereas the vanilla binarized GCN yields the result of 0.669 ± 0.070 in MAE, which is far behind that of the full-precision one.

We explore the reason behind this challenge of implausible performance, by looking into the internal computational process in binarized GNNs. Specifically, we look back on Fig. 3, which shows the example workflow at a binarized GCN layer where the feature transformation is performed before the aggregation step. It is noticeable that the result of 1-bit operations lies in the discrete integer domain. The resulted feature space is thereby much smaller than that of the 32-bit floating-point operations. In other words, the outputs of 1-bit operations are less distinguishable from each other. This property, when appearing in the graph domain, leads to difficulties to extract and discriminate graph topologies in the neighborhood aggregation process, which in fact is the key to the success of graph networks.

To further illustrate this dilemma, we demonstrate a couple of examples in Fig. 4, including both max and mean aggregation schemes that are commonly leveraged in GNNs. Fig. 4(a) shows the aggregation results of the 32-bit GNN layer, where both of max and mean aggregators successfully distinguish the two different topological structures, respectively. However, for the aggregation results of discrete integer features in binarized GNNs (Fig. 4(b)), neither max nor mean aggregators can discriminate the corresponding two graph structures. Moreover, the situation will be more challenging for the case where the aggregation happens before the transformation, since the features fed into the aggregator are limited to only 1 or -1.

Nevertheless, from Fig. 4(b), we also found that, by combining different aggregation schemes, various graph topologies could in fact become distinguishable. This observation motivates us to develop possible solutions to alleviate the aforementioned dilemma in vanilla binarized GNNs. Specifically, we propose a couple of straightforward mixed multi-aggregators that combine the benefits of various aggregation schemes in two different ways. The first one conducts multiple times of message aggregation with several different aggregators and then computes the sum over the aggregation results, leading to the performance of 0.599 ± 0.017 in MAE with five standard aggregators. The second one, on the other hand, concatenates the results from several independent aggregators, achieving the average result of 0.614 ± 0.045 over 25 runs.

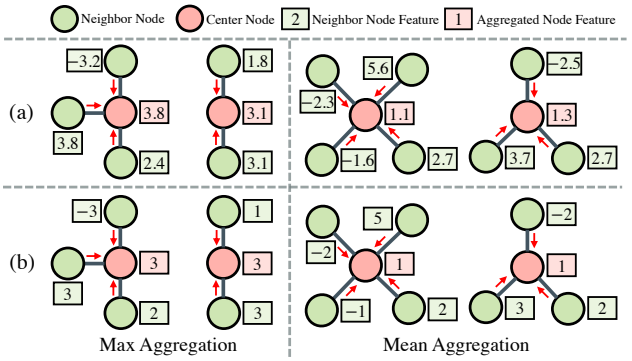

Figure 4. Example aggregation results of (a) conventional 32-bit GNN layer and (b) binarized GNN layer, corresponding to Fig. 3. For (a), both mean and max aggregators can distinguish the two graph structures; however, for binarized GNN (b), max and mean aggregators fail to differentiate between two topologies.

In spite of the improved performance, the devised possible solutions need to perform multiple times of feature aggregations at each GNN layer, resulting in heavy computational burdens. Motivated by this limitation, we introduce the proposed meta neighborhood aggregators, which aim to enhance the discriminative capability of topological structures and meanwhile enjoy efficient computations.

4. Meta Neighborhood Aggregation

4.1. Overview

Towards addressing the aforementioned limitations of the devised mixed multi-aggregators, we introduce in this section the proposed concept of the *Meta Aggregator*, which aims to adaptively and efficiently adjust the way to aggregate information in a learnable manner. Towards this end, we propose a couple of specific forms of meta aggregators, *i.e.*, the *exclusive* meta aggregation method and the *diffused* meta aggregation method, as illustrated in Fig. 5.

The exclusive form, termed as *Greedy Gumbel Neighborhood Aggregator* (GNA), learns to determine a single optimal aggregation scheme from a pool of candidate aggregators, according to the individual characteristics of the quantized graph features, as shown in the upper part of Fig. 5. The diffused meta form, on the other hand, adap-

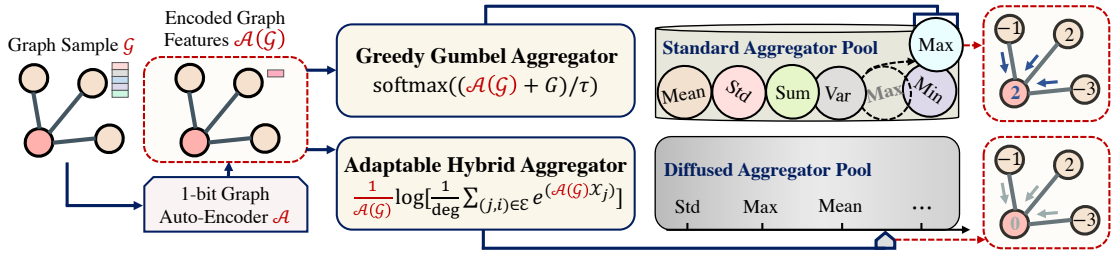

Figure 5. The overall framework of the proposed meta neighborhood aggregation methods. The upper row illustrates the workflow of the exclusive meta aggregator GNA, which receives the encoded graph features from the binarized graph auto-encoder $\mathcal{A}$ (*i.e.*, the pink trapezoid) and exclusively determines a single optimal layer-wise and node-wise aggregator from a candidate aggregator pool. The lower row, on the other hand, demonstrates the diffused meta aggregator ANA, which amalgamates various aggregation behaviors.

tively learns a customized aggregation formulation that can potentially incorporate the properties of several independent aggregators, thereby termed as *Adaptable Hybrid Neighborhood Aggregator* (ANA) shown in the lower part of Fig. 5.

In what follows, we detail the devised two forms of meta neighborhood aggregation methods, *i.e.*, GNA and ANA, and also the associated training strategy.

4.2. Greedy Gumbel Aggregator

Motivated by the observation from Fig. 4, where different single aggregators work for a corresponding set of cases as explained in Sect. 3, we propose the idea of adaptively determining the optimal aggregator depending on the specific input graphs, as depicted in the upper part of Fig. 5.

To this end, there are a few challenges to be addressed. First, the aggregation selector should understand the underlying characteristics of various input graphs without introducing much additional computational cost. To address this issue, we propose to leverage a 1-bit graph auto-encoder to extract meaningful information from input graphs, which is then exploited to guide the decision of different aggregation methods.

The second challenge is how to incorporate the discrete selections into the gradient descent process in training GNNs. One straightforward solution would be to model the discrete determination process as a state classification problem and to consider the various aggregators in the candidate pool as different labels. However, this naïve attempt does not account for the uncertainty of the selector, which is likely to cause the model collapse problem where the output choice is independent of the input graphs, such as always or never picking up a specific aggregator.

To alleviate this dilemma, we propose to impose stochasticity in the aggregator decision process with greedy Gumbel sampling [28, 45] and propagate gradients through stochastic neurons through the continuous form of Gumbel-Max trick [18]. Specifically, we introduce such stochasticity by greedily sampling noise from the Gumbel distribution, due to its property of Gumbel-Max trick [9]. In terms of Gumbel random variables, the Gumbel-Max trick

Algorithm 1 Training a lightweight 1-bit GNN model with the proposed meta neighborhood aggregators.

Input: L: the number of layers; $\mathcal{W}$: the GNN model weight; $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$: input graph data with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$; $\mathcal{X}$: the input binarized node feature vector; $\mathcal{A}$: the graph auto-encoder; *Meta-Aggre.*$\in$\{GNA, ANA\}: the choice of meta neighborhood aggregators.

Output: $\mathcal{M}_b$: Target 1-bit binarized GNN model.

for $\ell = 1$ to L **do**
  Feed the graph sample $\mathcal{G}$ into the GNN layer ℓ;
  Binarize the GNN weight $\mathcal{W}^\ell$ into $\mathcal{W}_b^\ell$ by Eq. 2;
  Perform 1-bit transformation with $\mathcal{X}$ and $\mathcal{W}_b^\ell$;
  Binarize the weight $\mathcal{W}^{\mathcal{A}^\ell}$ of $\mathcal{A}^\ell$ into $\mathcal{W}_b^{\mathcal{A}^\ell}$ by Eq. 2;
  Obtain the encoded features $\mathcal{A}^\ell(\mathcal{G})$ with $\mathcal{W}_b^{\mathcal{A}^\ell}$;
  // Identify the choice from the two meta aggregators
  if *Meta-Aggre.* is GNA **then**
    // Exclusively decide an optimal aggregator
    Feed $\mathcal{A}^\ell(\mathcal{G})$ into the GNA module.
    Obtain the decision GNA_i^ℓ for node v_i by Eq. 4;
    Perform aggregations with the obtained GNA_i^ℓ;
  else if *Meta-Aggre.* is ANA **then**
    // Generate a diffused aggregator
    Feed $\mathcal{A}^\ell(\mathcal{G})$ into the ANA module;
    Obtain the diffused aggregator ANA_i^ℓ by Eq. 5;
    Perform aggregations with the obtained ANA_i^ℓ;
  end if
end for
Optimize the binarized GNN $\mathcal{M}_b$ for epochs by Eq. 3.

can be utilized to parameterize discrete distributions. However, there is a argmax operation in the Gumbel-Max trick, which is not differentiable. We thereby resort to its continuous relaxation form, termed as Gumbel-softmax estimator, to address this issue, which uses a softmax function to replace the undifferentiable argmax function.

With the aforementioned graph auto-encoder and also the Gumbel-softmax estimator to address the two challenges, respectively, the proposed greedy Gumbel aggrega-

Table 1. Results on the ZINC dataset with different architectures, in terms of the mean absolute error (MAE). From left to right: the results of the full-precision GNNs (Full), those of the 1-bit GNNs without the proposed meta aggregators (Vanilla), and the results of the 1-bit GNNs with GNA and ANA. We also provide the p-value of the paired t-test to demonstrate the statistically meaningful improvements by the proposed GNA and ANA.

Methods	Full (GAT) [46]	Vanilla (GAT) [14]	**GNA (GAT)**	**ANA (GAT)**	Full (GCN) [21]	Vanilla (GCN) [14]	**GNA (GCN)**	**ANA (GCN)**
Bit-width	32/32	1/1	**1/1**	**1/1**	32/32	1/1	**1/1**	**1/1**
Param Size	399.941KB	81.7070KB	**82.0610KB**	**81.8799KB**	402.645KB	82.2002KB	**82.5566KB**	**82.3740KB**
Test MAE±SD	0.476±0.006	0.670±0.064	**0.592±0.013**	**0.566±0.012**	0.407±0.018	0.669±0.070	**0.608±0.024**	**0.607±0.020**
Train MAE±SD	0.300±0.024	0.610±0.066	**0.531±0.013**	**0.453±0.019**	0.303±0.026	0.624±0.069	**0.558±0.027**	**0.564±0.021**
***p*-value**	GNA vs. Vanilla: 3.010×10^{-7} / ANA vs. Vanilla: 2.359×10^{-10}				GNA vs. Vanilla: 1.597×10^{-4} / ANA vs. Vanilla: 9.787×10^{-5}			

tor (GNA) for node v_i can then be formulated as:

$$\text{GNA}_i^\ell = \text{softmax}\Big(\big(\mathcal{A}^\ell(\mathcal{G}) + G\big)/\tau\Big), \tag{4}$$

where $\mathcal{A}^\ell$ represents the binarized graph auto-encoder at layer ℓ that extracts principal and meaningful information, and G denotes the sampled Gumbel random noise. $\mathcal{G}$ is the input subgraph with one centered node v_i and a set of its neighboring nodes v_j where the connection $(v_i, v_j) \in \mathcal{E}$. τ is a constant that denotes the temperature of the softmax. GNA_i^ℓ is the output one-hot vector that indicates the aggregator decision at node v_i and layer ℓ from a pool of candidate aggregators like $\{\text{max}, \text{min}, \text{std}, \text{var}, \ldots, \text{mean}\}$.

In this way, the proposed greedy Gumbel aggregator adaptively decides the optimal aggregator conditioned on each specific node and layer in a learnable manner, which can significantly improve the topological discriminative capability of the vanilla binary GNN model.

4.3. Adaptable Hybrid Aggregator

Despite the improved representational ability, the performance of the greedy Gumbel aggregator is bottlenecked by that of the existing standard aggregators, which leaves room for further improvement. Motivated by this observation, we further devise an adaptable hybrid neighborhood aggregator (ANA) that can generate a hybrid form of the several standard aggregators in a learnable manner, thereby simultaneously retaining the advantages of different aggregators. The overall computational pipeline of ANA is demonstrated in the lower part of Fig. 5.

We start by giving the developed graph-based mathematical formulation for diffused message aggregation, defined as follows:

$$\text{ANA}_i^\ell = \frac{1}{\mathcal{A}^\ell(\mathcal{G})} \log\left[\frac{1}{\text{deg}_i} \sum_{(j,i)\in\mathcal{E}} \exp(\mathcal{A}^\ell(\mathcal{G})\, \mathcal{X}_j^\ell)\right], \tag{5}$$

where deg_i is the in-degree of the node v_i and $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is the graph sample with edges $(v_i, v_j) \in \mathcal{E}$. We use $\mathcal{A}^\ell$ to denote the 1-bit graph auto-encoder at layer ℓ, as is also used in Eq. 4. $\mathcal{X}_j^\ell$ represents the feature vector of the neighboring node v_j at layer ℓ, whereas ANA_i^ℓ is the obtained diffused aggregator.

Eq. 5 can essentially approximate the max and mean functions, depending on the output of graph auto-encoder $\mathcal{A}^\ell(\mathcal{G})$. Specifically, higher $\mathcal{A}^\ell(\mathcal{G})$ will lead to a behavior similar to that of the max aggregator, while smaller values of $\mathcal{A}^\ell(\mathcal{G})$ generate an effect of the mean neighborhood aggregation. Detailed mathematical proof is provided in the supplementary material.

By slightly changing the form of Eq. 5, we can also approximate other aggregators. For example, by simply adding a minus to the input graph features, Eq. 5 can approach the behavior of the min aggregation. Also, by utilizing the fact $\text{Var}(\mathcal{X}) = \text{mean}(\mathcal{X}^2) - \big(\text{mean}(\mathcal{X})\big)^2$, the variance aggregator can be approximated by adding the square operations to Eq. 5. More detailed derivations and mathematical proofs can be found in the supplement.

Furthermore, it is also possible to simultaneously combine the benefits of all these approximated aggregators, by summing multiple terms in Eq. 5 with graph-based learnable weighting factors that adaptively control the diffused degree of various aggregator approximations. We illustrate the corresponding sophisticated formulation and also more detailed explanations in the supplementary material.

4.4. Training Strategy

We also propose a training strategy, tailored for the proposed method. As a whole, the principal operations of training a 1-bit GNN model with the proposed meta neighborhood aggregation approaches is concluded in Alg. 1. For the sake of clarity, we omit the bias terms in our illustration, which have similar behavior to that of the GNN weight $\mathcal{W}$. Also, we take the case where the feature transformation happens before the aggregation step as an example to illustrate the overall workflow.

As can be observed from Alg. 1, at each layer, the input graph is fed into the lightweight 1-bit graph auto-encoder $\mathcal{A}$ to extract useful information that is beneficial to the following meta aggregators. Followed by this graph encoding process, the meta neighborhood aggregation module receives the encoded features and exclusively determines an optimal aggregator, or produces a diffused aggregator that

Table 2. Results of the proposed meta aggregation methods and other approaches for 32-bit full-precision models on the ZINC dataset, in terms of MAE. The results are averaged over 25 independent runs with 25 different random seeds.

Methods	Param Size	Test MAE±SD	Train MAE±SD
GatedGCN [3]	413.027KB	0.426±0.012	0.272±0.023
GraphSage [10]	371.004KB	0.475±0.007	0.296±0.030
GIN [55]	402.652KB	0.387±0.019	0.319±0.020
MoNet [31]	414.070KB	0.386±0.009	0.299±0.016
GCN [21]	402.645KB	0.407±0.018	0.303±0.026
GAT [46]	399.941KB	0.476±0.006	0.300±0.024
GNA (Ours)	**411.270KB**	**0.337±0.021**	**0.160±0.026**
ANA (Ours)	**404.504KB**	**0.325±0.015**	**0.109±0.014**

amalgamates the behaviors of several independent aggregators. The desired 1-bit GNN model can eventually be obtained by optimizing the model for epochs with the straight-through estimator, as explained in Sect. 3.

5. Experiments

In this section, we perform extensive experiments on three publicly available benchmarks across diversified problem domains, including graph regression, node classification, and 3D object recognition. Followed by the evaluations, we further provide detailed discussions regarding the strengths and weaknesses of the devised meta aggregators.

5.1. Experimental Settings

Datasets. We validate the effectiveness of the proposed meta aggregation methods on three different datasets, each of which specializes in a distinct task. Specifically, for the task of graph regression, we use the ZINC dataset [16], which is one of the most popular real-world molecular datasets [7]. The goal of ZINC is to regress a specific molecular property, *i.e*. the constrained solubility, which is a critical property for developing GNNs for molecules [61].

Also, for the node classification task, we adopt the protein-protein interaction (PPI) dataset [64], which is a multi-label dataset with 24 graphs corresponding to different human tissues. Each node in the PPI dataset is labeled with various protein functions. The objective of PPI is thereby to predict the 121 protein functions from the interactions of human tissue proteins. Furthermore, we utilize ModelNet40 [54] for the evaluation on the task of 3D object classification. ModelNet40 is a popular dataset for 3D object analysis [33, 34], containing 12,311 meshed CAD models from 40 shape categories in total. Each object comprises a set of 3D points, with the 3D coordinates as the features. The goal is to predict the category of each 3D shape.

Implementation Details. We primarily use three heterogeneous architectures, including graph convolutional network (GCN) [21], graph attention network (GAT) [46], as well as dynamic graph convolutional model (DGCNN) [52] to evaluate the proposed meta aggregation approach. For other settings such as learning rates and batch size, we follow those in the works of [7], [46], and [52] for the tasks of graph regression, node classification, and point cloud classification, respectively.

In particular, for more convincing evaluations, we report the results on the ZINC dataset over 25 independent runs with 25 different random seeds. Also, as done in the field of CNN binarization [37], we keep the first and the last GNN layer full-precision and binarize the other GNN layers for all the comparison methods. More detailed task-by-task architecture designs as well as the hyperparameter settings can be found in the supplementary material.

Table 3. Results on the PPI dataset for the task of node classification, in terms of micro-averaged F_1 score. Detailed network architectures can be found in the supplementary material.

Methods	Bit-width	Param Size	F_1 Score
Full Prec. [46]	32/32	43.7712MB	98.70
Vanilla [14]	1/1	28.2560MB	92.68
GNA (Ours)	**1/1**	**28.2572MB**	**97.52**
ANA (Ours)	**1/1**	**28.2565MB**	**97.71**

5.2. Results

Graph Regression. Tab. 1 shows the ablation results of the vanilla 1-bit GNN models and those of GNNs with the proposed meta neighborhood aggregators GAN and ANA. Specifically, we report the results on two GNN architectures, *i.e*., GCN [21] and GAT [46], by averaging over 25 independent runs with 25 seeds.

The proposed GNA and ANA, as shown in Tab. 1, achieves gratifying performance in terms of both test and train MAE, and at the same time maintains a compact model size. Moreover, we provide in the last row of Tab. 1 the p-value of the paired t-test between the 1-bit GNNs with a fixed aggregator (Vanilla) and those with the proposed learnable meta aggregators. The corresponding results statistically validate the effectiveness of the proposed method.

Furthermore, we show in Tab. 2 the results of extending the proposed meta aggregators to full-precision GNNs and compare them with those of the state-of-the-art approaches [3, 10, 55, 31, 21, 46]. Specifically, the results in the last two rows of Tab. 2 are obtained by simply replacing the pre-defined aggregator in GAT with the proposed GNA and ANA. As can be observed from Tab. 2, the proposed method outperforms other approaches by a large margin, and meanwhile introduces few additional parameters.

Node Classification. In Tab. 3, we demonstrate the results of different methods with the GAT architecture. The proposed GNA and ANA, as shown in Tab. 3, yield results on par with those of the 32-bit full-precision models, but comes

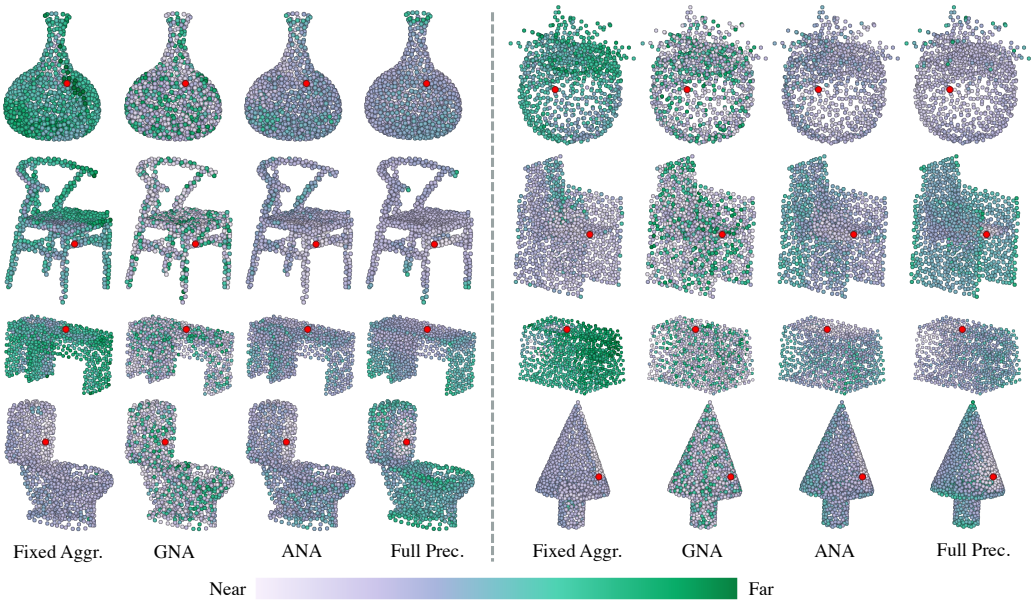

Figure 6. Visualization results of the learned feature space, depicted as the distance between the red point and the rest of the others. The visualized features are extracted from the intermediate layer of the models. More results can be found in the supplementary material.

Table 4. Results on the ModelNet40 dataset for 3D object recognition, in terms of the overall accuracy (Acc) and the mean class accuracy (mAcc).

Methods	Bit-width	Param Size	Acc (%)	mAcc (%)
Full Prec. [52]	32/32	1681.66KB	92.42	89.51
Vanilla [14]	1/1	1091.20KB	74.19	65.95
GNA (Ours)	**1/1**	**1091.30KB**	**78.36**	**71.67**
ANA (Ours)	**1/1**	**1091.30KB**	**84.64**	**78.89**

with a more lightweight architecture. The proposed method also outperforms the vanilla 1-bit GNN model that relies on a fixed aggregation scheme.

3D Object Recognition. The results of the proposed approach and other methods on the ModelNet40 dataset are shown in Tab. 4. We build our network here based on the architecture designed in [58]. We also demonstrate in Fig. 6 the corresponding visualization results of different approaches, where the column termed as "Fixed Aggr." in Fig. 6 corresponds to the "Vanilla" model in Tab. 4. With the proposed meta aggregation schemes, the 1-bit GNN model gains a boost by more than 10% in both the overall accuracy and the mean class accuracy. This improvement is also illustrated in Fig. 6, where the proposed meta aggregators help the 1-bit GNN learn a closer structure to that of the full-precision GNN model.

5.3. Discussions

We provide here a detailed account of the strengths and weaknesses of the proposed two meta aggregators GNA and ANA. For the exclusive meta form GNA, the performance can potentially be further enhanced with the advance of novel aggregation schemes. In other words, the results of GNA depend on those of every single aggregator in the candidate aggregation pool, which at the same time is a weakness of GNA since its performance is bottlenecked by that of the single aggregator. The diffused form ANA, on the other hand, may simultaneously retain the benefits of several popular aggregators. However, the mathematical form in Eq. 5 limits the type of aggregators that ANA can potentially approximate, meaning that ANA may not have much room for further improvement even with the emergence of novel and prevailing aggregators in the future.

6. Conclusions

In this paper, we propose a couple of learnable aggregation schemes for 1-bit compact GNNs. The goal of the proposed method is to enhance the topological discriminative ability of the 1-bit GNNs. This is achieved by adaptively selecting a single aggregator, or generating a hybrid aggregation form that can simultaneously maintain the strengths of several aggregators. Moreover, the proposed meta aggregation schemes can be readily extended to the full-precision GNN models. Experiments across various domains demonstrate that, with the proposed meta aggregators, the 1-bit GNN yields results on par with those of the cumbersome full-precision ones. In our future work, we will strive to generalize the proposed aggregator to compact and lightweight visual transformers.

Acknowledgements. Mr Yongcheng Jing is supported by ARC FL-170100117. Xinchao Wang is supported by the Start-up Fund of National University of Singapore.

References

[1] Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. Binary graph neural networks. *arXiv preprint arXiv:2012.15823*, 2020. 3

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[3] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017. 7

[4] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. In *BMVC*, 2019. 3

[5] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *CVPR*, 2020. 2

[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: training deep neural networks with binary weights during propagations. In *NeurIPS*, 2015. 3

[7] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020. 1, 2, 4, 7

[8] Jiayuan Gu, Han Hu, Liwei Wang, Yichen Wei, and Jifeng Dai. Learning region features for object detection. In *ECCV*, 2018. 1, 2

[9] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. US Government Printing Office, 1954. 5

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017. 2, 7

[11] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018. 1, 2

[12] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020. 1

[13] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *NeurIPS*, 2018. 2

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NeurIPS*, 2016. 3, 6, 7, 8

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 3

[16] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 2012. 4, 7

[17] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016. 2

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 5

[19] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Amalgamating knowledge from heterogeneous graph neural networks. In *CVPR*, 2021. 2

[20] Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016. 3

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 1, 2, 3, 6, 7

[22] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 1, 2

[23] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019. 3

[24] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018. 2

[25] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *AAAI*, 2018. 2

[26] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. In *AAAI*, 2021. 2

[27] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, 2019. 2

[28] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *NeurIPS*, 2014. 5

[29] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *ICLR*, 2020. 3

[30] Stefan Milz, Georg Arbeiter, Christian Witt, Bassam Abdallah, and Senthil Yogamani. Visual slam for automated driving: Exploring the applications of deep learning. In *CVPR Workshop*, 2018. 1

[31] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017. 7

[32] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019. 2

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2, 7

[34] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2, 7

[35] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *ECCV*, 2018. 2

[36] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *ICCV*, 2017. 1, 2

[37] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 2020. 7

[38] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 3

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 1986. 3

[40] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 1, 2

[41] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *TNN*, 2008. 2

[42] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. In *AAAI*, 2019. 2

[43] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo, and Mingli Song. Progressive network grafting for few-shot knowledge distillation. In *AAAI*, 2021. 2

[44] Chengchao Shen, Mengqi Xue, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation. In *ICCV*, 2019. 2

[45] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. *IJCV*, 2020. 5

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018. 1, 2, 6, 7

[47] Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, Xiangjian He, Yiguang Lin, and Xuemin Lin. Binarized graph neural network. *arXiv preprint arXiv:2004.11147*, 2020. 3

[48] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*, 2018. 2

[49] Junfu Wang, Yunhong Wang, Zhen Yang, Liang Yang, and Yuanfang Guo. Bi-gcn: Binary graph convolutional network. *arXiv preprint arXiv:2010.07565*, 2020. 3

[50] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR Workshop*, 2019. 3

[51] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019. 1

[52] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019. 2, 3, 7, 8

[53] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *TNNLS*, 2020. 1

[54] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 7

[55] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019. 1, 3, 7

[56] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 2

[57] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. In *NeurIPS*, 2020. 2

[58] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *CVPR*, 2020. 1, 8

[59] Yiding Yang, Zhou Ren, Haoxiang Li, Chunluan Zhou, Xinchao Wang, and Gang Hua. Learning dynamics via graph neural networks for human pose estimation and tracking. In *CVPR*, 2021. 1, 2

[60] Yiding Yang, Xinchao Wang, Mingli Song, Junsong Yuan, and Dacheng Tao. Spagan: Shortest path graph attention network. In *IJCAI*, 2019. 1

[61] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS*, 2018. 7

[62] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *CVPR*, 2017. 2

[63] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018. 1, 2

[64] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 2017. 7