Hierarchical Graph Capsule Network

Jinyu Yang[1][*], **Peilin Zhao**[2], **Yu Rong**[2], **Chaochao Yan**[1], **Chunyuan Li**[1], **Hehuan Ma**[1], **Junzhou Huang**[1]

[1]University of Texas at Arlington
[2]Tencent AI Lab
jzhuang@uta.edu

Abstract

Graph Neural Networks (GNNs) draw their strength from explicitly modeling the topological information of structured data. However, existing GNNs suffer from limited capability in capturing the hierarchical graph representation which plays an important role in graph classification. In this paper, we innovatively propose hierarchical graph capsule network (HGCN) that can jointly learn node embeddings and extract graph hierarchies. Specifically, disentangled graph capsules are established by identifying heterogeneous factors underlying each node, such that their instantiation parameters represent different properties of the same entity. To learn the hierarchical representation, HGCN characterizes the part-whole relationship between lower-level capsules (part) and higher-level capsules (whole) by explicitly considering the structure information among the parts. Experimental studies demonstrate the effectiveness of HGCN and the contribution of each component. Code: https://github.com/uta-smile/HGCN

Introduction

GNNs (Scarselli et al. 2008), especially graph convolutional networks (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015) have demonstrated remarkable performance in modeling structured data in a wide variety of fields, such as social networks (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Li et al. 2019) and graph-based representations of molecules (Gilmer et al. 2017; Rong et al. 2020a). The common practice is to recursively update node embeddings by aggregating (or message passing) information from topological neighbors such that the GNNs can capture the local structure of nodes. Subsequently, the learned embeddings can be used in downstream analyses, e.g., node classification (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Huang et al. 2018; Xu et al. 2018b; Rong et al. 2020b), link prediction (Zhang and Chen 2018), and graph classification (Duvenaud et al. 2015; Dai, Dai, and Song 2016; Gilmer et al. 2017). However, those GNNs fail to capture the hierarchical representations of graphs (Ying et al. 2018b), which is essential for many scenarios. For instance, in order to predict the properties of a given molecule, it would be highly desirable

to infer the sub-parts which are important for the molecular properties hierarchically.

To this end, various graph pooling methods are recently proposed, aiming to learn the coarse-grained graph structure by either reserving the most informative nodes (Gao and Ji 2019; Lee, Lee, and Kang 2019) or aggregating nodes belonging to the same cluster (Ying et al. 2018b; Yuan and Ji 2020; Khasahmadi et al. 2020; Wang et al. 2020; Bianchi, Grattarola, and Alippi 2020). In particular, the latter attracts considerable attention mainly attributed to its remarkable performance. Such kind of methods learn a cluster assignment matrix to map each node to a set of clusters that may correspond to strongly connected communities within a social network or functional groups within a molecule. However, their limitations lie in that (i) simply grouping node features fails to effectively model the part-whole relationship that is crucial in characterizing the hierarchical structure, and (ii) they ignore the entanglement of the latent factors behind node embeddings, resulting in limited capacity in preserving detailed node/graph properties and modeling graph hierarchy. For example, it is of particular importance to consider the interaction of heterogeneous factors (e.g., work, hobby) underlying each node, in order to identify the communities in a social network.

Capsule neural networks (CapsNets) have proved its effectiveness in modeling hierarchical relationships on image data by exploiting the fact that while viewpoint changes have complicated effects on pixel intensities, they have linear effects at the part/object level (Sabour, Frosst, and Hinton 2017; Hinton, Sabour, and Frosst 2018; Kosiorek et al. 2019). In contrast to convolutional neural networks, CapsNets use activity vectors or pose matrices to represent the entities. Moreover, the viewpoint-invariant relationship between the part and the whole is characterized by trainable transformation matrices, which is under the assumption that the human visual system relies on parse tree-like structure to recognize objects. Such representations make CapsNets especially appealing in reasoning the part-whole hierarchy and robust to adversarial attacks (Hinton, Sabour, and Frosst 2018; Qin et al. 2020). However, how to effectively take advantage of CapsNets to benefit graph classification remains largely unexplored.

In this work, we present the hierarchical graph capsule network (HGCN) that is able to jointly learn node embeddings and extract the hierarchical structure of the input graph.

Specifically, to preserve detailed node/graph information, we build graph capsules by disentangling heterogeneous factors behind the node embeddings such that each capsule encodes different properties of the same entity. In order to capture the graph hierarchy, multiple graph capsule layers are stacked to get coarser and coarser representations. In each layer, (i) to infer the votes of instantiation parameters of higher-level capsules (wholes), we propose transformation GNNs to reason about the part-whole relationship by explicitly considering the structure information among lower-level capsules (parts); (ii) each of these votes that are weighted by a routing weight, are iteratively routed to the potential wholes that correspond to tight clusters in the votes. We further introduce the auxiliary graph reconstruction to enhance the representation capacity of graph capsules and the training stability. As a consequence, HGCN is capable of modeling hierarchical representations of the input graph and benefits the goal of graph classification.

Our main contributions can be summarized as: (i) we propose a novel capsule-based graph neural network to learn node embeddings and hierarchical graph representations simultaneously, (ii) we demonstrate the effectiveness of considering the entanglement of latent factors and the structure information within the parts in modeling part-whole relationships on the graph data; (iii) comprehensive empirical studies demonstrate that our method achieves remarkably superior improvement over the state-of-the-art approaches on 11 commonly used benchmarks.

Related Work

Graph Neural Networks GNNs attempt to exploit the structure information underlying graph structured data in order to benefit various downstream tasks (Li et al. 2016; Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Li et al. 2018; Xu et al. 2018b; Luan et al. 2019; Rong et al. 2020b; You, Ying, and Leskovec 2019). Recent studies have proved GNNs' wide applicability in, for example, drug discovery (Yan et al. 2020; Gilmer et al. 2017; Ma et al. 2020), protein interface prediction (Fout et al. 2017), and recommendation system (Ying et al. 2018a). Current GNNs can be mainly summarized as two streams: spectral-based methods and spatial-based methods. The spectral-based methods largely rely on the convolution operation defined in the Fourier domain with spectral filters (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015). This kind of method is further simplified and extended by introducing polynomial spectral filters (Defferrard, Bresson, and Vandergheynst 2016) or linear filters (Kipf and Welling 2017). To deal with arbitrarily structured graphs, the spatial-based methods define convolutions directly on the graph by aggregating features from topological neighbors (Atwood and Towsley 2016; Niepert, Ahmed, and Kutzkov 2016; Hamilton, Ying, and Leskovec 2017; Vaswani et al. 2017).

Graph Pooling It is widely recognized that pooling operation plays an important role in graph classification (Errica et al. 2020) which requires the graph level representation. The most straightforward way is, to sum up, or take an average of all node features (Hamilton, Ying, and Leskovec 2017;

Xu et al. 2018a). The limitation of such strategy is that the hierarchical information which is crucial in capturing graph structure is not considered. Inspired by the downsampling in convolutional neural networks, recent studies propose to adaptively keep the most informative nodes in a hierarchical manner (Gao and Ji 2019; Lee, Lee, and Kang 2019), or aggregate maximal cliques by only using topological information (Luzhnica, Day, and Lio 2019). Another line of work focuses on finding strongly connected communities on a graph. This is typically achieved by learning a cluster assignment matrix in order to map each node to a set of clusters (Ying et al. 2018b; Ranjan, Sanyal, and Talukdar 2020; Yuan and Ji 2020; Khasahmadi et al. 2020). Most recent studies approach this problem by leveraging more advanced clustering techniques, such as local graph Fourier transform (Ma et al. 2019b), spectral clustering (Bianchi, Grattarola, and Alippi 2020), and compressive Haar transforms (Wang et al. 2020). However, simply grouping node features has limited capacity in modeling the part-whole relationships, especially for biological data. In this work, we reason about the part-whole hierarchy by exploring the interaction of underlying latent factors and structure information among the parts, then use an routing mechanism to assign the parts to wholes.

Capsule Networks A capsule (Hinton, Krizhevsky, and Wang 2011) is a group of neurons whose orientation represents the instantiation parameters such as pose (position, size) of an entity (e.g., an object). The probability that the entity exists can be represented by the capsule length (Sabour, Frosst, and Hinton 2017) or a logistic unit (Hinton, Sabour, and Frosst 2018; Kosiorek et al. 2019). Compared to a single neuron, a capsule contains different properties of the same entity and can preserve hierarchical relationships between lower-level capsules (e.g., eyes, mouth) and higher-level capsules (e.g., face). Such part-whole relationships are described by trainable transformation matrices which are viewpoint-invariant. Concretely, a lower-level capsule (part) makes predictions for the pose of each higher-level capsule (whole) by multiplying its own pose by the transformation matrices. Routing-by-agreement is then performed between two adjacent capsule layers to update the probability with which a part is assigned to a whole (Sabour, Frosst, and Hinton 2017; Hinton, Sabour, and Frosst 2018).

Inspired by this, GCAPS-CNN (Verma and Zhang 2018) builds capsules on graphs by considering higher-order statistical moments as instantiation parameters. Normal graph convolution is then carried out to aggregate information from neighbors and the covariance is computed as the permutation invariant feature for graph classification. Its most obvious drawback lies in that the hierarchical structure of the graph is not considered. Different from GCAPS-CNN, we explicitly take into account the hierarchy between two consecutive capsule layers through trainable transformation GNNs. Caps-GNN (Xinyi and Chen 2019) uses multiple GNN channels to build graph capsules and follow the same voting strategy as (Sabour, Frosst, and Hinton 2017) to predict higher-level capsules. However, simply using the transformation matrix ignores the local structure information among lower-level capsules and fails to describe part-whole relationships in

graphs. Our method introduces transformation GNNs to reason about the pose of each whole in the layer above, which is in contrast to each individual part making its own prediction. A further advantage of transformation GNNs is that they save orders of magnitude of memory compared to transformation matrices used in previous work (Xinyi and Chen 2019; Sabour, Frosst, and Hinton 2017; Hinton, Sabour, and Frosst 2018). Furthermore, different from CapsGNN that reconstructs the histogram of input nodes, we reconstruct the adjacency matrix of the input graph to ensure the quality of graph capsules and enhance the training stability.

Preliminaries

Graph Classification A graph G with N nodes is represented as $(\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix, and $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the node feature matrix with feature dimension d. Given a set of labeled graphs $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), ...\}$, the goal of graph classification is to learn a mapping $f : \mathcal{G} \to \mathcal{Y}$, where $G_i \in \mathcal{G}$ and $y_i \in \mathcal{Y}$. For example, each graph is a molecule, and its label indicates whether it is toxic.

Graph Neural Networks To extract useful information from local neighborhoods, our method is built upon GNNs by following the general "message-passing" paradigm, which is formulated as:

$$\mathbf{H}^{(l+1)} = \mathcal{M}(\mathbf{A}, \mathbf{H}^{(l)}; \mathbf{W}^{(l)}), \qquad (1)$$

where $\mathcal{M}$ indicates the message passing function with various possible implementations (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017), $\mathbf{W}^{(l)}$ is learnable weight matrix, $\mathbf{H}^{(l+1)}$ and $\mathbf{H}^{(l)}$ are the node embeddings of layer $l+1$ and l, respectively. The input node embeddings $\mathbf{H}^{(1)}$ are initialized using the node feature $\mathbf{X}$, i.e., $\mathbf{H}^{(1)} = \mathbf{X}$. The final node representations are denoted by $\text{GNN}(\mathbf{A}, \mathbf{X}) = \mathbf{H}^{(L_{\text{GNN}})} \in \mathbb{R}^{N \times h}$ with L_{GNN} iterations.

Methodology

In this section, we begin by first briefing the proposed HGCN as shown in Figure 1, then we detail each component in the following sections. The goal of HGCN is to jointly learn node embeddings and coarsen the graph through exploiting hierarchical information. To achieve this, we disentangle node representations to build the graph capsule by considering heterogeneous factors underlying each edge connection. Therefore, each graph capsule is composed of multiple independent latent factors that represent different properties of the same entity. To learn hierarchical graph representations, transformation GNNs (TGNNs) are proposed to encode the part-whole relationship between lower-level and higher-level graph capsules. Specifically, a capsule in one layer votes for the instantiation parameters of each capsule in the layer above through TGNNs which highly depend on the structure information of lower-level capsules. Each of these votes is then routed to a higher-level capsule that receives a cluster of similar votes by a routing-by-agreement strategy. To encourage the graph capsules to encode the instantiation parameters of the input graph and also enhance the training stability,

we further introduce the auxiliary graph reconstruction to reconstruct the input adjacency matrix.

Disentangled Graph Capsules

In most cases, highly complex interactions are involved in the connection between each node pair in a graph. For example, the edges between a node and its neighbors in a social network are driven by heterogeneous factors, since a person connects with others for various reasons such as exercise, work, etc. Therefore, it is necessary to disentangle the explanatory factors of variations underlying the node representations. Furthermore, each node embedding is considered as multiple individual scalar features in existing GNNs, which are proved to have limited capability in preserving the graph properties (Verma and Zhang 2018; Xinyi and Chen 2019).

To address these two limitations, motivated by (Sabour, Frosst, and Hinton 2017), we propose graph capsules to describe the given graph. Specifically, we disentangle the latent factors of each node embedding and use the disentangled node representation to represent graph capsules (Figure 1A). In this way, each graph capsule is composed of multiple heterogeneous factors, and each factor describes a specific instantiation parameter of the entity/node. Formally, given $G = (\mathbf{A}, \mathbf{X})$, the node i is denoted by $\mathbf{x}_i \in \mathbb{R}^d$. We project the input node features into K different subspaces, assuming that there are K latent factors/instantiation parameters:

$$\mathbf{z}_{i,k} = \sigma(\mathbf{W}_k^T \mathbf{x}_i) + \mathbf{b}_k, \qquad (2)$$

where $\mathbf{W}_k \in \mathbb{R}^{d \times \frac{h}{K}}$ and $\mathbf{b}_k \in \mathbb{R}^{\frac{h}{K}}$ are learnable parameters, σ is a nonlinear activation function, and $\frac{h}{K}$ is the dimension of each factor. Although more sophisticated implementations of node disentanglement are possible (Ma et al. 2019a), we use linear projection in our study attributed to its efficiency and remarkable performance. Therefore, each graph capsule is represented by a pose matrix $\mathbf{Z}_i \in \mathbb{R}^{K \times \frac{h}{K}}$ (Hinton, Sabour, and Frosst 2018). For simplicity, we reshape $\mathbf{Z}_i$ to the vector format $\mathbf{z}_i \in \mathbb{R}^h$. Recall that the existence probability of an entity represented by a capsule is measured by the capsule length (Sabour, Frosst, and Hinton 2017), we thus squash $\mathbf{z}_i$ as follows:

$$\mathbf{p}_i = squash(\mathbf{z}_i) = \frac{\|\mathbf{z}_i\|^2}{1 + \|\mathbf{z}_i\|^2} \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|}, \qquad (3)$$

where $\mathbf{u}_i^{(1)} = \mathbf{p}_i \in \mathbb{R}^h$ is the primary graph capsule representing the lowest level of entities, such as atoms in the molecular graph.

Hierarchical Capsule Layers

To obtain hierarchical graph representation, it is essential to capture the part-whole relationship between adjacent capsule layers. Such relationship is measured by viewpoint-invariant transformation matrix $\mathbf{T}_{i,j}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ for each pair of lower-level capsule $\mathbf{u}_i^{(l)}$ and higher-level capsule $\mathbf{u}_j^{(l+1)}$ in previous studies (Sabour, Frosst, and Hinton 2017; Hinton, Sabour, and Frosst 2018), where d_l and d_{l+1} are the capsule dimensions of $\mathbf{u}_i^{(l)}$ and $\mathbf{u}_j^{(l+1)}$, respectively. However, $\mathbf{T}_{i,j}^{(l)}$ totally ignores
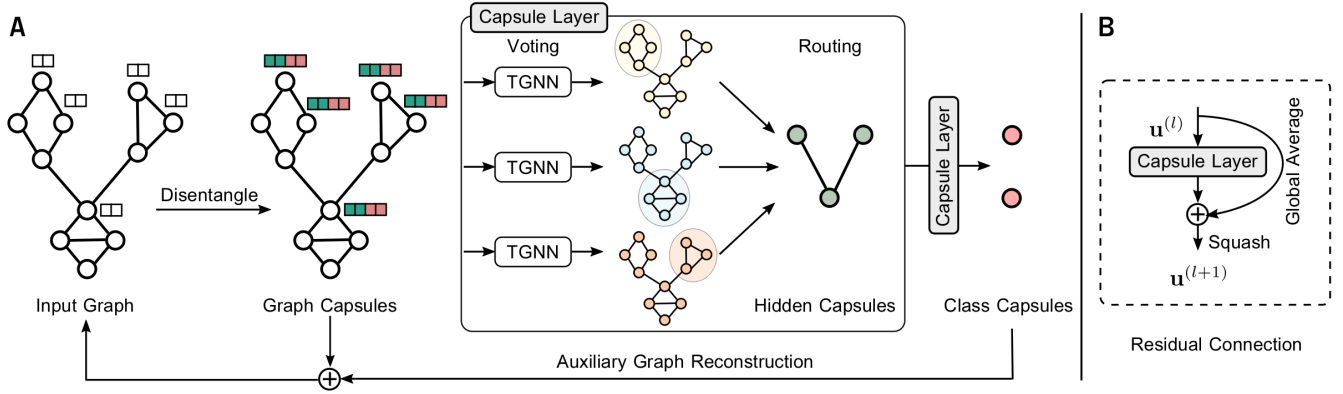
Figure 1: An overview of the proposed framework. (A) Given an input graph, we build graph capsules by learning disentangled node representations in order to take into account the heterogeneous factors behind each node. TGNNs are established to characterize the part-whole relationship, and a routing strategy is used to predict higher-level capsules that receive a cluster of similar votes. (B) The residual connection that combines fine, low layer information with coarse, high layer information.
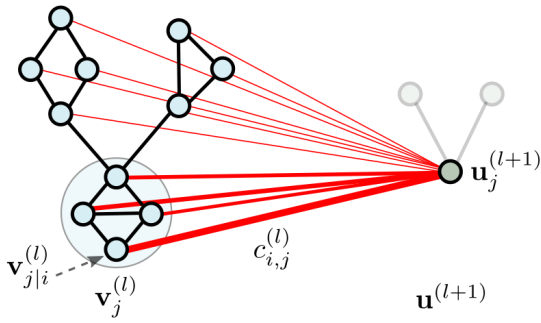

Figure 2: Cluster by agreement.

the structure information within $\mathbf{u}^{(l)}$, which is especially problematic for graph structured data. Furthermore, $\mathbf{T}^{(l)} \in \mathbb{R}^{N_l \times N_{l+1} \times d_l \times d_{l+1}}$ is extremely memory-consuming for the scenario where a large number of high-dimensional capsules are required.

To overcome these difficulties, we propose the transformation GNNs (TGNNs) to vote for the instantiation parameters of higher-level graph capsules (**voting**). When multiple votes agree, a higher-level capsule that receives a cluster of similar pose votes becomes active (**routing**). More concretely, we denote the graph capsules at layer l as $\mathbf{u}^{(l)} \in \mathbb{R}^{N_l \times d_l}$, the capsule number as N_l, and the adjacency matrix as $\mathbf{A}^{(l)}$. Our goal is to decide which capsules to activate in $\mathbf{u}^{(l+1)} \in \mathbb{R}^{N_{l+1} \times d_{l+1}}$ and how to assign each active lower-level capsule $\mathbf{u}_i^{(l)}$ to one active higher-level capsule $\mathbf{u}_j^{(l+1)}$. In practice, we set $N_{l+1} < N_l$ in order to get coarser and coarser graph representations (Figure 1A).

Voting For all capsules in $\mathbf{u}^{(l)}$, their poses are transformed by TGNNs to cast votes for the pose of each capsule in $\mathbf{u}^{(l+1)}$ by the following equation,

$$\mathbf{v}_j^{(l)} = \text{TGNN}_j(\mathbf{A}^{(l)}, \mathbf{u}^{(l)}), \qquad (4)$$

where $\mathbf{v}_j^{(l)} \in \mathbb{R}^{N_l \times d_{l+1}}$. Specifically, $\mathbf{v}_{j|i}^{(l)} \in \mathbb{R}^{d_{l+1}}$ is the vote for the pose of $\mathbf{u}_j^{(l+1)}$ predicted by the capsule $\mathbf{u}_i^{(l)}$. Note, TGNNs are learned discriminatively and could learn to represent part-whole relationships by considering the structure information of capsules in $\mathbf{u}^{(l)}$. This is different from previous studies that use one transformation matrix for each pair of $(\mathbf{u}_i^{(l)}, \mathbf{u}_j^{(l+1)})$. Compared to transformation matrices, TGNNs also save N_l orders of magnitude memory.

Routing Each of these votes is then weighted by an routing weight $c_{i,j}^{(l)}$ with which a part is assigned to a whole, where $c_{i,j}^{(l)} \geqslant 0$ and $\sum_{j=1}^{N_{l+1}} c_{i,j}^{(l)} = 1$. Here, $c_{i,j}^{(l)}$ is iteratively updated using an "routing-by-agreement" mechanism such that each vote in $\mathbf{v}^{(l)}$ is routed to a capsule in $\mathbf{u}^{(l+1)}$ that receives a cluster of similar votes (Figure 2). Formally, $c_{i,j}^{(l)}$ is defined as $c_{i,j}^{(l)} = \exp(b_{i,j}^{(l)})/\sum_k \exp(b_{i,k}^{(l)})$, where $b_{i,j}^{(l)}$ is initialized as $b_{i,j}^{(l)} = 0$. To iteratively search for the vote cluster, in each iteration we have,

$$\mathbf{u}_j^{(l+1)} = squash(\sum_i c_{i,j}^{(l)} \mathbf{v}_{j|i}^{(l)}) \qquad (5)$$

where $\mathbf{u}_j^{(l+1)}$ is the predicted capsule j in layer $l+1$, representing a tight cluster of votes from layer l. We update $b_{i,j}^{(l)}$ with $b_{i,j}^{(l)} = b_{i,j}^{(l)} + a_{i,j}^{(l)}$, where $a_{i,j}^{(l)} = \mathbf{v}_{j|i}^{(l)} \cdot \mathbf{u}_j^{(l+1)}$ indicates the agreement between each vote and vote cluster. It is worth mentioning that such top-down feedback also has a beneficial effect on the aggregation in the proposed TGNNs, such that TGNNs can more focus on aggregating information from neighbors that are likely to be in the same cluster. After R iterations, we get higher-level graph capsules $\mathbf{u}^{(l+1)}$ and the coarsened adjacency matrix defined as $\mathbf{A}^{(l+1)} = \mathbf{C}^{(l)^T} \mathbf{A}^{(l)} \mathbf{C}^{(l)} \in \mathbb{R}^{N_{l+1} \times N_{l+1}}$. As opposed to generating structurally independent higher-level capsules in previous work (Sabour, Frosst, and Hinton 2017; Xinyi and

Chen 2019), the capsule layer we developed is able to explicitly preserve the structure information which is encoded in $\mathbf{A}^{(l+1)}$.

Drawing inspiration from (Long, Shelhamer, and Darrell 2015), we add a residual connection at each pair of consecutive capsule layers, aiming to provide fine-grained information to higher-level capsules (Figure 1B). Formally, we have $\mathbf{u}^{(l+1)} \leftarrow \mathbf{u}^{(l+1)} + \text{GA}(\mathbf{u}^{(l)})$, where GA indicates the global average operation.

By stacking multiple capsule layers, we get the class capsules $\mathbf{u}^{(L)} \in \mathbb{R}^{O \times d_L}$ which are intended to encode feature attributes corresponding to the class, where O is the number of graph categories. The classification loss is measured by a margin loss (Sabour, Frosst, and Hinton 2017) which is formulated as:

$$\mathcal{L}_m(\mathbf{A}, \mathbf{X}) = \sum_{o \in O} [T_o \max(0, m^+ - \|\mathbf{u}_o^L\|)^2 + \lambda(1 - T_o) \max(0, \|\mathbf{u}_o^L\| - m^-)^2], \quad (6)$$

where $m^+ = 0.9$, $m^- = 0.1$, $T_o = 1$ iff the input graph has label o, and λ is ued to stop the initial learning from shrinking the length of class capsules.

Auxiliary Graph Reconstruction

To encourage the graph capsules to encode the instantiation parameters of the input graph and to improve the training stability, we introduce a reconstruction loss to constraint the capsule reconstruction to closely match the class-conditional distribution. Specifically, we first mask out all but the winning capsule (the capsule corresponds to ground truth) and combine them with primary capsules by following the equation,

$$\mathbf{Z} = \mathbf{u}^{(1)} + (\mathbf{W}_r^T \Phi(\mathbf{u}^{(L)}) + \mathbf{b}_r), \quad (7)$$

where $\mathbf{Z} \in \mathbb{R}^{N \times d_1}$, Φ is the mask operation, $\mathbf{W}_r \in \mathbb{R}^{(O \times d_L) \times d_1}$, and $\mathbf{b}_r \in \mathbb{R}^{d_1}$. The reconstruction loss is then defined as,

$$\mathcal{L}_r(\mathbf{A}, \mathbf{X}) = -\frac{1}{N^2} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{c \in \{0,1\}} \mathbf{A}^{(j,k,c)} log(\tilde{\mathbf{A}}^{(j,k,c)}), \quad (8)$$

where $\tilde{\mathbf{A}} = \mathbf{Z}\mathbf{Z}^T$ is the reconstructed adjacency matrix of the input graph. Taken together, we reach the optimization objective of our method as $\min_{\theta} \sum_{G \in \mathcal{D}} \mathcal{L}_m(\mathbf{A}, \mathbf{X}) + \beta \mathcal{L}_r(\mathbf{A}, \mathbf{X})$, where θ are all learnable parameters, and β leverages the importance of $\mathcal{L}_r$. The whole training process is detailed in Algorithm 1.

Experiments

In this section, we conduct empirical studies on 11 benchmark datasets and demonstrate HGCN's superiority over a number of state-of-the-art graph classification methods. Extensive ablation studies are also performed to evaluate the effectiveness of each component in our model.

Algorithm 1: Training process with K latent factors, L capsule layers, and R iterations of routing.

Input: $G = (\mathbf{A}, \mathbf{X}), \mathbf{A} \in \mathbb{R}^{N \times N}, \mathbf{X} \in \mathbb{R}^{N \times d}$
Result: class capsules $\mathbf{u}^{(L)}$
for $i \leftarrow 1$ **to** N **do**
  for $k \leftarrow 1$ **to** K **do**
    $\mathbf{z}_{i,k} = \sigma(\mathbf{W}_k^T \mathbf{x}_i) + \mathbf{b}_k$ // K latent factors
  end
end
$\mathbf{u}_i^{(1)} = squash(\mathbf{z}_i)$ // disentangled graph capsules
for $l \leftarrow 1$ **to** L **do**
  $b_{i,j}^{(l)} = 0$
  for $j \leftarrow 1$ **to** N_{l+1} **do**
    $\mathbf{v}_j^{(l)} = \text{TGNN}_j(\mathbf{A}^{(l)}, \mathbf{u}^{(l)})$ // votes
  end
  for $r \leftarrow 1$ **to** R **do**
    $c_{i,j}^{(l)} = \exp(b_{i,j}^{(l)})/\sum_k \exp(b_{i,k}^{(l)})$
    $\mathbf{u}_j^{(l+1)} = squash(\sum_i c_{i,j}^{(l)} \mathbf{v}_{j|i}^{(l)})$
    $b_{i,j}^{(l)} = b_{i,j}^{(l)} + \mathbf{v}_{j|i}^{(l)} \cdot \mathbf{u}_j^{(l+1)}$
  end
  $\mathbf{A}^{(l+1)} = \mathbf{C}^{(l)T} \mathbf{A}^{(l)} \mathbf{C}^{(l)} \in \mathbb{R}^{N_{l+1} \times N_{l+1}}$
end
return $\mathbf{u}^{(L)} \in \mathbb{R}^{O \times d_L}$ // class capsules

---

Datasets Eleven commonly used benchmarks including (i) seven biological graph datasets, i.e., MUTAG, NCI1, PROTEINS, D&D, ENZYMES, PTC, NCI109; and (ii) four social graph datasets, i.e., COLLAB, IMDB-Binary (IMDB-B), IMDB-Multi (IMDB-M), Reddit-BINARY (RE-B), are used in this study. It is noteworthy that the social graphs have no node attributes, while the biological graphs come with categorical node attributes. More details about the data statistics and properties can be found in Supplementary.

Baseline Methods We compare with two capsule-based methods, i.e., CapsGNN (Xinyi and Chen 2019) and GCAPS-CNN (Verma and Zhang 2018). We also conduct a comparison with a number of state-of-the-art GNN-based methods, including PATCHY-SAN (PSCN) (Niepert, Ahmed, and Kutzkov 2016), PSCN (Niepert, Ahmed, and Kutzkov 2016), GCN (Kipf and Welling 2017), Deep Graph CNN (DGCNN) (Zhang et al. 2018), CLIQUEPOOL (Luzhnica, Day, and Lio 2019), DIFFPOOL (Ying et al. 2018b), ASAP (Ranjan, Sanyal, and Talukdar 2020), SAGPool (Lee, Lee, and Kang 2019), EigenPooling (Ma et al. 2019b), GIN (Xu et al. 2018a), GFN (Chen, Bian, and Sun 2019), HaarPool (Wang et al. 2020), STRUCTPOOL (Yuan and Ji 2020), and MemGNN/GMN (Khasahmadi et al. 2020). For kernel-based methods, we consider WL (Shervashidze et al. 2011), DGK (Yanardag and Vishwanathan 2015), AWE (Ivanov and Burnaev 2018), and GK (Shervashidze et al. 2009).

Experimental Settings We set $K = 4$, $R = 3$, $\lambda = 0.5$, $\beta = 0.1$, $L = 2$, and follow the same settings in previous studies (Ying et al. 2018b) to perform 10-fold cross-

Algorithm		MUTAG	NCI1	PROTEINS	D&D	ENZYMES	PTC	NCI109
AWE	Kernel	87.87±9.76	—	—	71.51±4.02	35.77±5.93	—	—
GK		81.58±2.11	62.49±0.27	71.67±0.55	78.45±0.26	32.70±1.20	59.65±0.31	62.60±0.19
WL		82.05±0.36	82.19±0.18	74.68±0.49	79.78±0.36	52.22±1.26	57.97±0.49	82.46±0.24
DGK		87.44±2.72	80.31±0.46	75.68±0.54	73.50±1.01	53.43±0.91	60.08±2.55	80.32±0.33
SAGPool	GNN	—	67.45±1.11	71.86±0.97	76.45±0.97	—	—	67.86±1.41
CLIQUEPOOL		—	—	72.59	77.33	60.71	—	—
ASAP		—	71.48±0.42	74.19±0.79	76.87±0.7	—	—	70.07±0.55
HaarPool		77.60±8.94	80.17±2.29	73.23±2.51	—	—	—	69.61±1.49
EigenPooling		79.50	77.00	76.60	78.60	64.50	—	74.90
DGCNN		85.83±1.66	74.44±0.47	75.54±0.94	79.37±0.94	51.00±7.29	58.59±2.47	75.03±1.72
PSCN		88.95±4.37	76.34±1.68	75.00±2.51	76.27±2.64	—	62.29±5.68	—
GIN		89.40±5.60	82.70±1.70	76.20±2.80	—	—	64.60±7.00	—
DIFFPOOL		—	—	76.25	80.64	62.53	—	74.10
GCN		87.20±5.11	83.65±1.69	75.65±3.24	79.12±3.07	66.50±6.91	—	70.70
GFN		90.84±7.22	82.77±1.49	76.46±4.06	78.78±3.49	70.17±5.58	—	—
CapsGNN	Caps	86.67±6.88	78.35±1.55	76.28±3.63	75.38±4.17	54.67±5.67	—	—
GCAPS-CNN		—	82.72±2.38	76.40±4.17	77.62±4.99	61.83±5.39	66.01±5.91	81.12±1.28
Ours		**93.16±6.10**	**84.87±1.68**	**77.99±3.16**	**80.99±2.58**	**78.00±4.89**	**66.54±7.97**	**83.91±1.27**

Table 1: Performance comparison on biological graphs. "Caps" indicates capsule-based GNNs.

Algorithm		COLLAB	IMDB-B	IMDB-M	RE-B
GK	Kernel	72.84±0.28	65.87±0.98	43.89±0.38	65.87±0.98
AWE		73.93±1.94	74.45±5.83	51.54±3.61	87.89±2.53
WL		79.02±1.77	73.40±4.63	49.33±4.75	81.10±1.90
DGK		73.09±0.25	66.96±0.56	44.55±0.52	78.04±0.39
PSCN	NN	72.60±2.15	71.00±2.29	45.23±2.84	86.30±1.58
DGCNN		73.76±0.49	70.03±0.86	47.83±0.85	76.02±1.73
DIFFPOOL		75.48	—	—	—
GCN		81.72±1.64	73.30±5.29	51.20±5.13	—
GFN		81.50±2.42	73.00±4.35	51.80±5.16	—
GIN		80.20±1.90	75.10±5.10	52.30±2.80	92.40±2.50
GCAPS-CNN	Caps	77.71±2.51	71.69±3.40	48.50±4.10	87.61±2.51
CapsGNN		79.62±0.91	73.10±4.83	50.27±2.65	—
Ours		**82.86±1.81**	**77.20±4.73**	**52.80±2.45**	**93.15±1.58**

Table 2: Performance comparison on social graphs.

validation for performance evaluation. For each dataset, we select a single epoch that has the best cross-validation accuracy averaged over the 10 folds, and report the average and standard deviation of test accuracies at the selected epoch. HaarPool (Wang et al. 2020) repeats each experiment 10 times with different random seeds, for a fair comparison, we run HaarPool with 10-fold cross-validation and report the result. STRUCTPOOL (Yuan and Ji 2020) and MemGNN/GMN (Khasahmadi et al. 2020) use a different assessment criterion that selects the epoch with the best test accuracy on each fold and then take the average. We name such assessment criterion as C^* and run our method by following the same paradigm for comparison. Unless otherwise indicated, we use the result reported in the original paper for other baseline methods. For TGNNs, we adopt the GCN (Kipf and Welling 2017) with $L_{GNN} = 1$.

Experimental Results We first compare HGCN with existing state-of-the-art graph classification methods on seven biological datasets. Our results demonstrate that HGCN achieves the best performance based on the widely used criterion (Table 1) and competes favorably against three C^*- based baselines (Supplementary Table 5). In particular, compared with two capsule-based GNNs (i.e., GCAPS-CNN and CapsGNN), we boost of 16.17%, 6.49%, 3.37%, and 2.79% improvement on ENZYMES, MUTAG, D&D, and NCI109, respectively. The reason is that higher-order statistical moments of local neighbors are used in GCAPS-CNN to build graph capsules, this strategy, however, fails to identify the underlying latent factors which are important in preserving node/graph property and extracting the hierarchical representations. Although this limitation can be partially alleviated by using multiple graph channels as reported in CapsGNN, the transformation matrices used in CapsGNN ignore the structure information involved in lower-level capsules. In contrast, we disentangle node representations to explicitly consider the entanglement of heterogeneous factors, and propose transformation GNNs to measure structure-aware part-whole relationships. Furthermore, it should be noted that we achieve the largest performance gains on ENZYMES which has six classes, compared to the rest of datasets with only two classes. This observation implies that HGCN is able to capture more complicated and accurate hierarchy for the multiclass classification problem than other methods. For instance, although assignment matrix-based methods (e.g., DIFFPOOL) are capable of mapping nodes to a set of clusters for structurally simple graphs, its representation power is limited to complex and crowded graphs such as ENZYMES. By contrast, our method performs iterative routing to obtain the cluster of agreement, which jointly learns the hierarchical graph representation and provides the necessary deprecation of assignment ambiguity (Sabour, Frosst, and Hinton 2017).

		MUTAG	NCI1	PROTEINS	D&D	ENZYMES	PTC	NCI109
A1	ablation	91.46±5.77	80.24±1.78	76.37±3.11	78.35±2.73	70.00±4.51	63.61±12.47	80.64±2.09
A2		92.08±5.10	**85.28±1.37**	77.63±3.03	80.64±3.65	77.00±5.82	64.24±8.45	83.84±1.41
A3		92.11±7.13	84.87±1.07	77.54±3.44	79.96±3.26	77.67±3.70	65.10±8.81	83.84±1.48
Ours		**93.16±6.10**	84.87±1.68	**77.99±3.16**	**80.99±2.58**	**78.00±4.89**	**66.54±7.97**	**83.91±1.27**
2	K	91.55±5.64	84.50±1.87	77.90±3.31	80.04±2.77	77.50±5.05	64.81±9.45	**84.15±1.93**
8		93.16±6.59	84.65±1.17	77.18±2.74	78.69±2.99	77.83±4.97	65.71±7.95	84.03±2.31
1	R	91.05±6.59	83.36±1.62	76.64±2.46	79.45±2.47	77.33±5.34	64.23±8.60	82.26±1.75
2		92.11±6.68	84.01±1.68	77.45±2.92	79.97±3.60	77.83±3.77	65.13±8.07	83.35±1.36
4		92.63±5.66	84.26±1.20	76.92±2.66	**81.15±3.79**	77.00±4.29	64.21±8.31	83.60±2.05
5		92.08±6.19	84.55±1.23	77.18±1.99	80.64±3.11	77.33±3.87	65.38±14.31	83.11±1.82
Ours		**93.16±6.10**	**84.87±1.68**	**77.99±3.16**	80.99±2.58	**78.00±4.89**	**66.54±7.97**	83.91±1.27

Table 3: Ablation studies (upper part) and sensitivity analyses (lower part) on biological graphs.

		COLLAB	IMDB-B	IMDB-M	RE-B
A1	ablation	81.44±1.57	64.70±11.65	51.00±3.10	91.45±2.13
A2		82.20±1.41	74.80±4.18	49.80±6.39	92.95±1.94
A3		**83.08±1.69**	75.90±4.36	52.00±1.54	92.70±1.53
Ours		82.86±1.81	**77.20±4.73**	**52.80±2.45**	**93.15±1.58**
2	K	82.94±1.66	75.60±6.69	52.20±2.46	92.85±2.12
8		**83.10±1.80**	74.90±5.82	51.67±3.99	**93.35±1.73**
1	R	82.52±1.87	74.50±4.58	51.87±3.23	92.45±1.82
2		82.36±2.03	74.80±4.73	51.40±3.68	92.65±1.73
4		83.04±1.92	74.70±2.87	51.67±3.07	93.10±0.91
5		82.64±1.44	74.50±6.26	51.13±2.93	92.85±1.67
Ours		82.86±1.81	**77.20±4.73**	**52.80±2.45**	93.15±1.58

Table 4: Ablation studies (upper part) and sensitivity analyses (lower part) on social graphs.

Table 2 and Table 6 (Supplementary) show the performance comparison on four social graph datasets, where the key challenge is to identify strongly connected communities. Similarly, we achieve the significant performance improvement over baseline models, indicating that HGCN can better reason about the part-whole relationships in social networks. This is also consistent with the fact that highly complex interactions are involved in social graphs, which can be modeled by identifying heterogeneous factors underlying each node. Most importantly, considering the entanglement of the latent factors enables more accurate hierarchy learning.

Ablation Studies Comprehensive ablation studies are carried out in this section to understand the contribution of each component (i.e., disentangled graph capsules, capsule layers, and auxiliary graph reconstruction) in our method. Specifically, we (i) directly use the input node representation to serve as graph capsules, without considering the entanglement of heterogeneous factors (A1); (ii) remove the residual connection between adjacent capsule layers (A2); and (iii) remove the auxiliary graph reconstruction (A3). The results illustrated in Table 3 and Table 4 (upper part) reveal that

(i) disentangling node representation allows us to characterize the latent factors underlying each node and in turn more accurately preserve the node/graph properties and capture the part-whole relationship; (ii) combining fine, low layer information with coarse, high layer information provides us an ability to enhance the final graph-level representation; and (iii) graph reconstruction plays an important role in encoding the instantiation parameters of the input graph and enhancing the training stability. Thus, we reach the conclusion that each component in our method is necessary and contributes to the performance improvement. One exception is NCI1 (A2) and COLLAB (A3), where residual connection and auxiliary graph reconstruction bring inferior performance which may be caused by overfitting.

Sensitivity Analyses In this section, we analyze the sensitivity of HGCN to the number of latent factors $K = \{2, 8\}$ and the number of routing iterations $R = \{1, 2, 4, 5\}$, where our method with setting: $K = 4$ and $R = 3$. As shown in Table 3 and Table 4 (lower part), the results demonstrate that HGCN is not very sensitive to these two hyper-parameters. Although $K = 8$ brings limited performance improvement on COLLAB and RE-B than $K = 4$ (ours), the computational complexity is doubled in calculating the disentangled graph capsules. Similarly, $R = 4$ requires more routing iterations, albeit with 0.16% accuracy boost on D&D.

Conclusion

In this paper, we introduce a novel HGCN framework for graph classification, which is able to explicitly extract hierarchical graph representations. Built upon disentangled graph capsules by identifying heterogeneous factors behind each node, HGCN encodes part-whole relationships by considering the structure information of lower-level parts and iteratively infer the pose of higher-level objects. Empirical studies demonstrate the superiority of our framework over existing graph classification methods on 11 commonly used benchmarks.

Acknowledgments

References

Atwood, J.; and Towsley, D. 2016. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*.

Bianchi, F. M.; Grattarola, D.; and Alippi, C. 2020. Spectral clustering with graph neural networks for graph pooling. *International Conference on Machine Learning* .

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations* .

Chen, T.; Bian, S.; and Sun, Y. 2019. Are powerful graph neural nets necessary? a dissection on graph classification. *International Conference on Learning Representations (Workshop)* .

Dai, H.; Dai, B.; and Song, L. 2016. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*.

Errica, F.; Podda, M.; Bacciu, D.; and Micheli, A. 2020. A fair comparison of graph neural networks for graph classification. *International Conference on Learning Representations* .

Fout, A.; Byrd, J.; Shariat, B.; and Ben-Hur, A. 2017. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*.

Gao, H.; and Ji, S. 2019. Graph u-nets. *International Conference on Machine Learning* .

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. *International conference on machine learning* .

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* .

Hinton, G. E.; Krizhevsky, A.; and Wang, S. D. 2011. Transforming auto-encoders. In *International conference on artificial neural networks*. Springer.

Hinton, G. E.; Sabour, S.; and Frosst, N. 2018. Matrix capsules with EM routing. *International Conference on Learning Representations* .

Huang, W.; Zhang, T.; Rong, Y.; and Huang, J. 2018. Adaptive Sampling Towards Fast Graph Representation Learning. *Advances in Neural Information Processing Systems* 31: 4558–4567.

Ivanov, S.; and Burnaev, E. 2018. Anonymous walk embeddings. In *International Conference on Machine Learning*, 2186–2195. PMLR.

Khasahmadi, A. H.; Hassani, K.; Moradi, P.; Lee, L.; and Morris, Q. 2020. Memory-Based Graph Networks. *International Conference on Learning Representations* .

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations* .

Kosiorek, A.; Sabour, S.; Teh, Y. W.; and Hinton, G. E. 2019. Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. *International Conference on Machine Learning* .

Li, J.; Rong, Y.; Cheng, H.; Meng, H.; Huang, W.; and Huang, J. 2019. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*.

Li, R.; Wang, S.; Zhu, F.; and Huang, J. 2018. Adaptive graph convolutional neural networks. In *Thirty-second AAAI conference on artificial intelligence*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2016. Gated graph sequence neural networks. *International Conference on Learning Representations* .

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Luan, S.; Zhao, M.; Chang, X.-W.; and Precup, D. 2019. Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks. In *Advances in neural information processing systems*.

Luzhnica, E.; Day, B.; and Lio, P. 2019. Clique pooling for graph classification. *International Conference on Learning Representations (Workshop)* .

Ma, H.; Bian, Y.; Rong, Y.; Huang, W.; Xu, T.; Xie, W.; Ye, G.; and Huang, J. 2020. Multi-View Graph Neural Networks for Molecular Property Prediction. *Advances in Neural Information Processing Systems (Workshop)* .

Ma, J.; Cui, P.; Kuang, K.; Wang, X.; and Zhu, W. 2019a. Disentangled graph convolutional networks. In *International Conference on Machine Learning*.

Ma, Y.; Wang, S.; Aggarwal, C. C.; and Tang, J. 2019b. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*.

Qin, Y.; Frosst, N.; Sabour, S.; Raffel, C.; Cottrell, G.; and Hinton, G. 2020. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. *International Conference on Learning Representations* .

Ranjan, E.; Sanyal, S.; and Talukdar, P. P. 2020. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. *Thirty-second AAAI conference on artificial intelligence* .

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020a. Self-Supervised Graph Transformer on Large-Scale Molecular Data. *Advances in Neural Information Processing Systems* 33.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020b. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* .

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* .

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *International Conference on Artificial Intelligence and Statistics*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*.

Verma, S.; and Zhang, Z.-L. 2018. Graph capsule convolutional neural networks. *International Conference on Machine Learning (Workshop)* .

Wang, Y. G.; Li, M.; Ma, Z.; Montufar, G.; Zhuang, X.; and Fan, Y. 2020. Haar graph pooling. *International Conference on Machine Learning* .

Xinyi, Z.; and Chen, L. 2019. Capsule graph neural network. *International Conference on Learning Representations* .

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018a. How powerful are graph neural networks? *International Conference on Learning Representations* .

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018b. Representation learning on graphs with jumping knowledge networks. *International conference on machine learning* .

Yan, C.; Ding, Q.; Zhao, P.; Zheng, S.; Yang, J.; Yu, Y.; and Huang, J. 2020. Retroxpert: Decompose retrosynthesis prediction like a chemist. *Advances in Neural Information Processing Systems* .

Yanardag, P.; and Vishwanathan, S. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018a. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018b. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems* .

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware graph neural networks. *International conference on machine learning* .

Yuan, H.; and Ji, S. 2020. STRUCTPOOL: STRUCTURED GRAPH POOLING VIA CONDITIONAL RANDOM FIELDS. *International Conference on Learning Representations* .

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.