NEURAL POOLING FOR GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tasks such as graph classification, require graph pooling to learn graph-level representations from constituent node representations. In this work, we propose two novel methods using fully connected neural network layers for graph pooling, namely Neural Pooling Method 1 and 2. Our proposed methods have the ability to handle variable number of nodes in different graphs, and are also invariant to the isomorphic structures of graphs. In addition, compared to existing graph pooling methods, our proposed methods are able to capture information from all nodes, collect second-order statistics, and leverage the ability of neural networks to learn relationships among node representations, making them more powerful. We perform experiments on graph classification tasks in the bio-informatics and social network domains to determine the effectiveness of our proposed methods. Experimental results show that our methods lead to an absolute increase of upto 1.2% in classification accuracy over previous works and a general decrease in standard deviation across multiple runs indicating greater reliability. Experimental results also indicate that this improvement in performance is consistent across several datasets.

1 INTRODUCTION

Over the past several years, there is a growing number of applications where data is generated from non-Euclidean domains and is represented as graphs with complex relationships and interdependency between entities. Deep learning generalised from grid-like data to the graph domain has led to the development of the remarkably successful Graph Neural Networks (GNNs) (Fan et al., 2019; Gao et al., 2019; Ma et al., 2019a; Wang et al., 2019b) and its numerous variants such the Graph Convolutional Network (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), graph attention network (GAT) (Veličković et al., 2018), jumping knowledge network (JK) (Xu et al., 2018), and graph isomorphism networks (GINs) (Xu et al., 2019), etc.

Pooling is a common operation in deep learning on grid-like data, such as images. Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. It reduces dimensionality and also provides local translational invariance. In the case of graph data, pooling is used to obtain a representation of a graph using its constituent node representations. However, it is challenging to develop graph pooling methods due to the some special properties of graph data such as the variable number of nodes in different graphs and the isomorphic structures of graphs. Firstly, the number of nodes varies in different graphs, while the graph representations are usually required to have the same fixed size to fit into other downstream machine learning models where they are used for tasks such as classification. Therefore, graph pooling should be capable of handling the variable number of node representations as inputs and producing fixed-sized graph representations. Secondly, unlike images and texts where we can order pixels and words according to the spatial structural information, there is no inherent ordering relationship among nodes in graphs. Therefore, isomorphic graphs should have the same graph representation, and hence, graph pooling should give the same output by taking node representations in any order as inputs

Our main contributions in this work are two novel graph pooling methods, Neural Pooling Method 1 and 2. These new pooling methods allow us to do the following,i) produce the same dimensional graph representation for graphs with variable number of nodes, ii) remain invariant to the isomorphic structures of graphs, iii) collect second- order statistics, iv) leverage trainable parameters in the form

of fully connected neural networks to learn relationships between underlying node representations to generate high quality graph representations which are then used for graph classification tasks.

Experiments are performed on two benchmark bio-informatics datasets and two popular social network datasets to demonstrate the effectiveness and superiority of our proposed graph pooling methods. Experimental results show that our methods lead to an improvement in classification accuracy over existing methods and are also more reliable as compared previous works.

2 Related Work

2.1 Graph Neural Networks

A graph can be represented by its adjacency matrix and node features. Formally, for a graph $\mathcal{G}$ consisting of n nodes, its topology information can be represented by an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and the node features can be represented as $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, assuming each node has a d-dimensional feature vector. GNNs learn feature representations for different nodes using these matrices (Gilmer et al., 2017). Several approaches are proposed to investigate deep GNNs, and they generally follow a neighborhood information aggregation scheme (Gilmer et al., 2017; Xu et al., 2019; Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018). In each step, the representation of a node is updated by aggregating the representations of its neighbors. Graph Convolutional Networks (GCNs) are popular variants of GNNs and inspired by the first order graph Laplacian methods (Kipf & Welling, 2017). Graph pooling is used to connect embedded graphs outputted by GNN layers with classifiers for graph classification. Given a graph, GNN layers produce node representations, where each node is embedded as a vector. Graph pooling is applied after GNN layers to process node representations into a single feature vector as the graph representation. A classifier takes the graph representation and performs graph classification.

2.2 Graph Pooling

Early studies employ simple methods such as averaging and summation as graph pooling (Xu et al., 2019; Duvenaud et al., 2015; Defferrard et al., 2016). However, averaging and summation do not capture the feature correlation information, curtailing the overall model performance (Zhang et al., 2018). Other studies have proposed advanced graph pooling methods, including DIFFPOOL (Ying et al., 2018), SORT-POOL (Zhang et al., 2018), TOPKPOOL (Gao & Ji, 2019), SAGPOOL (Lee et al., 2019), and EIGENPOOL (Ma et al., 2019b), and achieve great performance on multiple benchmark datasets. EIGENPOOL involves the computation of eigenvectors, which is slow and expensive. DIFFPOOL (Ying et al., 2018) treats the graph pooling as a node clustering problem. A cluster of nodes from the original graph are merged to form a new node in the new graph. DIFFPOOL (Ying et al., 2018) proposes to perform the graph convolution operation on node features to obtain node clustering assignment matrix. Intuitively, the class assignment of a given node should depend on the class assignments of other neighbouring nodes. However, DIFFPOOL does not explicitly consider high-order structural relationships, which we that are important for graph pooling. SORTPOOL (Zhang et al., 2018), TOPKPOOL (Gao & Ji, 2019), and SAGPOOL (Lee et al., 2019) learn to select important nodes from the original graph and use these nodes to build a new graph. They share the similar idea to learn a sorting vector based on node representations, which indicates the importance of different nodes. Then only the top k important nodes are selected to form a new graph while the other nodes are ignored. However, the ignored nodes may contain important features and this information is lost during pooling. It is worth noting that all the graph pooling methods mentioned till now only collect first-order statistics (Boureau et al., 2010). A recent study has proposed second order graph pooling methods $SOPool_{bimap}$ and $SOPool_{attention}$ (Wang & Ji, 2020). In this work, we propose two novel methods using fully connected neural network layers for graph pooling, namely Neural Pooling Method 1 and 2. Compared to existing graph pooling methods, our proposed methods are able to capture information from all nodes, collect second-order statistics, and leverage the ability of neural networks to learn relationships among node representations, making them more powerful.

3 METHODOLOGY

3.1 PROPERTIES OF GRAPH POOLING

Consider a graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ represented by its adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes in $\mathcal{G}$ and d is the dimension of node features. The node features may come from node labels or node degrees. Graph neural networks are known to be powerful in learning good node representation matrix $\boldsymbol{H}$ from $\boldsymbol{A}$ and $\boldsymbol{X}$:

$$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n]^T = \text{GNN}(\boldsymbol{A}, \boldsymbol{X}) \in \mathbb{R}^{n \times f} \tag{1}$$

where rows of $\boldsymbol{H}$, $\boldsymbol{h}_i \in \mathbb{R}^f$, $i = 1, 2, ..., n$, are representations of n nodes, and f is the dimension of the node representation obtained from the GNN and depends on the architecture of the GNN. The task that we focus on in this work is to obtain a graph representation vector $\boldsymbol{h}_G$ from $\boldsymbol{H}$, which is then fed into a classifier to perform graph classification:

$$\boldsymbol{h}_G = \text{g}([\boldsymbol{A}], \boldsymbol{H}) \in \mathbb{R}^c \tag{2}$$

where g(·) is the graph pooling function and c is the dimension of $\boldsymbol{h}_G$. Here, $[\boldsymbol{A}]$ means that the information from $\boldsymbol{A}$ can be optionally used in graph pooling. For simplicity, we omit it in the following discussion.

Note that the function g(·) must satisfy two requirements to serve as graph pooling. First, g(·) should be able to take $\boldsymbol{H}$ with variable number of rows as the inputs and produce fixed-sized outputs. Specifically, different graphs may have different number of nodes, which means that n is a variable. On the other hand, c, which is the dimension of the graph representation $\boldsymbol{h}_G$ is supposed to be fixed to fit into the classifier. Second, g(·) should output the same $\boldsymbol{h}_G$ when the order of rows of $\boldsymbol{H}$ changes. This permutation invariance property is necessary to handle isomorphic graphs. To be concrete, if two graph $\mathcal{G}_1 = (\boldsymbol{A}_1, \boldsymbol{X}_1)$ and $\mathcal{G}_2 = (\boldsymbol{A}_2, \boldsymbol{X}_2)$ are isomorphic, GNNs will output the same multi set of node representations. That is, there exists a permutation matrix $\boldsymbol{P} \in \{0,1\}^{n \times n}$ such that $\boldsymbol{H}_1 = \boldsymbol{P}\boldsymbol{H}_2$, for $\boldsymbol{H}_1 = \text{GNN}(\boldsymbol{A}_1, \boldsymbol{X}_1)$ and $\boldsymbol{H}_2 = \text{GNN}(\boldsymbol{A}_2, \boldsymbol{X}_2)$. However, the graph representation computed by g(·) should be the same, i.e., $\text{g}(\boldsymbol{H}_1) = \text{g}(\boldsymbol{H}_2)$ if $\boldsymbol{H}_1 = \boldsymbol{P}\boldsymbol{H}_2$.

3.2 NEURAL POOLING METHOD 1

Our first proposed method is called Neural Pooling Method 1. Consider a graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ represented by its adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes in $\mathcal{G}$ and d is the dimension of node features. The node features may come from node labels or node degrees. Graph neural networks are known to be powerful in learning good node representation matrix $\boldsymbol{H}$ from $\boldsymbol{A}$ and $\boldsymbol{X}$:

$$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n]^T = \text{GNN}(\boldsymbol{A}, \boldsymbol{X}) \in \mathbb{R}^{n \times f} \tag{3}$$

where rows of $\boldsymbol{H}$, $\boldsymbol{h}_i \in \mathbb{R}^f$, $i = 1, 2, ..., n$, are representations of n nodes, and f depends on the architecture of the GNN.

$\boldsymbol{H}$ is passed through a Fully Connected Neural Network Layer (FCL^1) to obtain $\boldsymbol{H}'$ as:

$$\boldsymbol{H}' = FCL^1(\boldsymbol{H}) \in \mathbb{R}^{n \times f'} \text{ where } f' < f \tag{4}$$

After this $\boldsymbol{H}'$ is again passed through a second Fully Connected Neural Network Layer (FCL^2) to obtain $\boldsymbol{Q}$ as:

$$\boldsymbol{Q} = FCL^2(\boldsymbol{H}') \in \mathbb{R}^{n \times 1} \tag{5}$$

Finally the graph representation $\boldsymbol{h}_G$ is obtained as:

$$\boldsymbol{h}_G = \boldsymbol{H}'^T \boldsymbol{Q} \in \mathbb{R}^{f' \times 1} \tag{6}$$

where $\boldsymbol{H}'^T$ denotes the transpose of $\boldsymbol{H}'$.

Neural Pooling Method 1 always outputs an f'-dimensional graph representation for $\boldsymbol{H} \in \mathbb{R}^{n \times f}$, regardless of the value of n. It is also invariant to permutation so that it outputs the same graph representation when the order of rows of $\boldsymbol{H}$ changes.

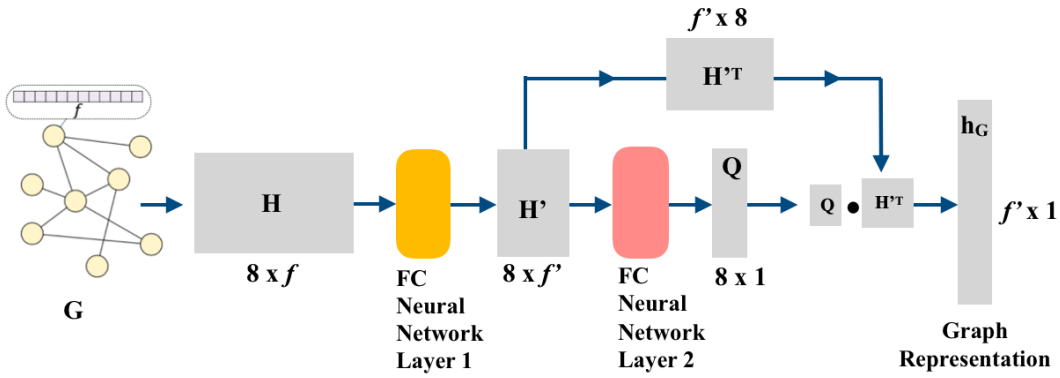

Figure 1: Illustration of our proposed Neural Pooling Method 1. This is an example for a graph $\mathcal{G}$ with 8 nodes. GNNs can learn representations for each node and graph pooling processes node representations into a graph representation vector.

Intuition: The FCL^1 performs the role of reducing the dimensionality of the input node representations. The trainable parameters of this FCL^1 can be thought of as learning a mapping from the f to the f'-dimensional space. The FCL^2 reduces the f'-dimensional node representations to a 1 dimensional representation, $\boldsymbol{Q}$. $\boldsymbol{H}' \in \mathbb{R}^{n \times f'}$ can be viewed as $\boldsymbol{H}' = [\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_{f'}]$, where $\boldsymbol{l}_j \in \mathbb{R}^n$, j=1, 2, ..., f'. The vector $\boldsymbol{l}_j$ encodes the spatial distribution of the j-th feature in the graph. Based on this view, $\boldsymbol{H}'^T\boldsymbol{Q}$ is able to capture the topology information and $\boldsymbol{Q}$ can be thought of as roughly encoding the position of nodes by learning the weights according to which the j-th feature is aggregated across the nodes.

Neural Pooling Method 1 hence, leverages the ability of neural networks to learn the topological structure as well as correlation among the node representations in $\boldsymbol{H}$, capturing essential features and connections between underlying data and in the process reducing the dimensionality of $\boldsymbol{H}$, resulting in an accurate representation of the input graph.

3.3 Neural Pooling Method 2

Our second proposed method is called Neural Pooling Method 2. Consider a graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ represented by its adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes in $\mathcal{G}$ and d is the dimension of node features. The node features may come from node labels or node degrees. Graph neural networks are known to be powerful in learning good node representation matrix $\boldsymbol{H}$ from $\boldsymbol{A}$ and $\boldsymbol{X}$:

$$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots\ldots\ldots, \boldsymbol{h}_n]^T = \text{GNN}(\boldsymbol{A}, \boldsymbol{X}) \in \mathbb{R}^{n \times f} \tag{7}$$

where rows of $\boldsymbol{H}$, $\boldsymbol{h}_i \in \mathbb{R}^f$, $i = 1, 2, \ldots, n$, are representations of n nodes, and f depends on the architecture of the GNN.

$\boldsymbol{H}$ is passed through a Fully Connected Neural Network Layer (FCL^1) to obtain $\boldsymbol{H}''$ as:

$$\boldsymbol{H}'' = FCL^1(\boldsymbol{H}) \in \mathbb{R}^{n \times f''} \text{ where } f'' < f \tag{8}$$

After this $\boldsymbol{H}''$ is again passed through a second Fully Connected Neural Network Layer (FCL^2) to obtain $\boldsymbol{H}'$ as:

$$\boldsymbol{H}' = FCL^2(\boldsymbol{H}'') \in \mathbb{R}^{n \times f'} \text{ where } f' < f'' \tag{9}$$

Finally the graph representation $\boldsymbol{h}_G$ is obtained as:

$$\boldsymbol{h}_G = \text{Flatten}(\boldsymbol{H}'^T\boldsymbol{H}') \in \mathbb{R}^{f'^2 \times 1} \tag{10}$$

where $\boldsymbol{H}'^T$ denotes the transpose of $\boldsymbol{H}'$.

Intuition: The FCL^1 performs the role of reducing the dimensionality of the input node representations. The trainable parameters of this FCL^1 can be thought of as learning a mapping from the f

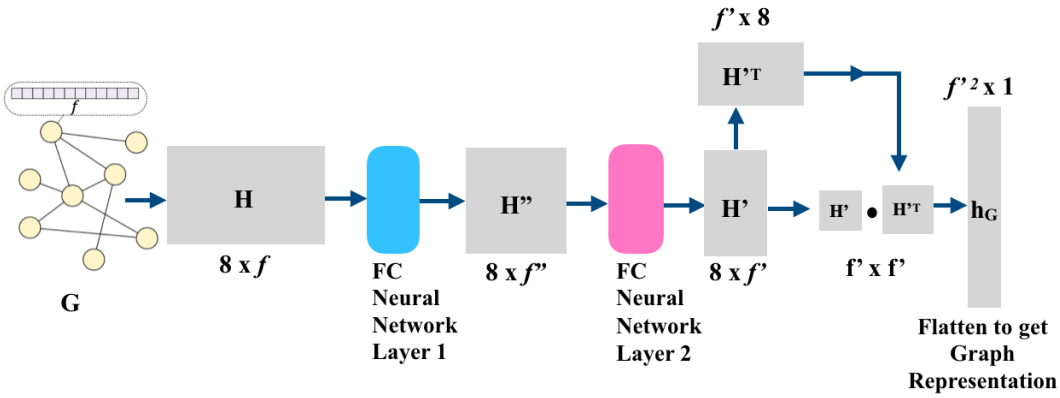

Figure 2: Illustration of our proposed Neural Pooling Method 2. This is an example for a graph $\mathcal{G}$ with 8 nodes. GNNs can learn representations for each node and graph pooling processes node representations into a graph representation vector.

to the f''-dimensional space. The FCL^2 further reduces the f''-dimensional node representations to a f' dimensional representation. $\boldsymbol{H}' \in \mathbb{R}^{n \times f'}$ can be viewed as $\boldsymbol{H}' = [\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_{f'}]$, where $\boldsymbol{l}_j \in \mathbb{R}^n$, j=1, 2, ..., f'. The vector $\boldsymbol{l}_j$ encodes the spatial distribution of the j-th feature in the graph. Based on this view, $\boldsymbol{H}'^T\boldsymbol{H}'$ is able to capture the topology information

Similar to the previous method, Neural Pooling Method 2 satisfies both the conditions of graph pooling which is that it always outputs an f'^2-dimensional graph representation for $\boldsymbol{H} \in \mathbb{R}^{n \times f}$, regardless of the value of n. It is also invariant to permutation so that it outputs the same graph representation when the order of rows of $\boldsymbol{H}$ changes.

4 Experimental Setup

We perform experiments on graph classification tasks in the bio-informatics and social network domains to demonstrate the effectiveness and superiority of our proposed methods, namely Neural Pooling Methods 1 and 2. The code used in our experiments is available at `https://github.com/sreesai1412/Neural-Pooling`. Details of datasets and parameter settings are described below.

4.1 Datasets

We use four graph classification datasets from (Yanardag & Vishwanathan, 2015), including two bioinformatics datasets (PTC, PROTEINS) and two social network datasets (IMDB-BINARY, IMDB-MULTI). Note that only bioinformatics datasets come with node labels. For the social network datasets, we use one-hot encoding of node degrees as features. The details of the datasets are summarized in Table 1.

- PTC (Toivonen et al., 2003) is a bioinformatics dataset of 344 graphs representing chemical compounds. Each node comes with one of 19 discrete node labels. The task is to predict the rodent carcinogenicity for each graph.
- PROTEINS (Borgwardt et al., 2005) is a bioinformatics dataset of 1,113 graph structures of proteins. Nodes in the graphs refer to secondary structure elements (SSEs) and have discrete node labels indicating whether they represent a helix, sheet or turn. And edges mean that two nodes are neighbors along the amino- acid sequence or in space. The task is to predict the protein function for each graph.
- IMDB-BINARY is a movie collaboration dataset of 1,000 graphs representing ego-networks for actors/actresses. The dataset is derived from collaboration graphs on Action and Romance genres. In each graph, nodes represent actors/actresses and edges simply

Table 1: Details of datasets

Name	PTC	PROTEINS	IMDB-B	IMDB-M
No. of graphs	344	1113	1000	1500
No. of classes	2	2	2	3
No. of nodes(max)	109	620	136	89
No. of nodes(avg.)	25.6	39.1	19.8	13.0

mean they collaborate the same movie. The graphs are labeled by the corresponding genre and the task is to identify the genre for each graph.

- IMDB-MULTI is multi-class version of IMDB- BINARY. It contains 1,500 ego-networks and has three extra genres, namely, Comedy, Romance and Sci-Fi.

4.2 Training and Evaluation

Following (Yanardag & Vishwanathan, 2015; Niepert et al., 2016), model performance is evaluated using 10-fold cross-validation and reported as the average and standard deviation of validation accuracies across the 10 folds. For GNNs, we follow the same training process in (Xu et al., 2019). The GNN has 5 layers. Each multi-layer perceptron (MLP) has 2 layers with batch normalization (Ioffe & Szegedy, 2015). Dropout (Srivastava et al., 2014) is applied in the classifiers. The Adam (Kingma & Ba, 2015) optimizer is used with the learning rate initialized as 0.01 and decayed by 0.5 every 50 epochs. The number of total epochs is selected according to the best cross-validation accuracy. We tune the number of hidden units (16, 32, 64) and the batch size (32, 128) using grid search.

4.3 Baselines

We compare our methods with various other graph pooling methods on the graph classification task, including DIFFPOOL (Ying et al., 2018), SORT-POOL (Zhang et al., 2018), TOPKPOOL (Gao & Ji, 2019), SAGPOOL (Lee et al., 2019), and EIGEN-POOL (Ma et al., 2019b). DIFFPOOL maps nodes to a pre-defined number of clusters but is hard to train. EIGENPOOL involves the computation of eigenvectors, which is slow and expensive. SORTPOOL, SAGPOOL and TOPKPOOL rely on the top-K sorting to select a fixed number (K) of nodes and order them, during which the information from unselected nodes is discarded. We also compare with some recent methods including COVPOOL (Wang et al., 2019a), ATTNPOOL (Girdhar & Ramanan, 2017) as well as second order pooling methods SOPool$_{bimap}$ and SOPool$_{attention}$ (Wang & Ji, 2020).

5 Results & Discussion

The results of our experiments are summarized in Table 2 .From the results we can see that our methods lead to an improvement in classification accuracy over existing methods and are also more reliable as compared previous works as observed from the lower values of standard deviation.This enhancement in performance is consistent across all the datasets. The results may be attributed to the fact that compared to existing graph pooling methods, our pooling methods are able to use information from all nodes, collect second-order statistics, and leverage the ability of neural networks to learn from underlying data, making them more powerful. The Neural Pooling methods utilize the ability of neural networks to learn the topological structure as well as correlation among the node representations in, capturing essential features and connections between underlying data.

6 Complexity

Consider a graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ represented by its adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes in $\mathcal{G}$ and d is the dimension of node features.

Table 2: Comparison results of our proposed methods with other graph pooling methods. Results shown are the average classification accuracy and standard deviation across 10-fold cross-validation.

	PTC	PROTEINS	IMDB-B	IMDB-M
SUM/AVG (Xu et al., 2019)	64.6 $\pm$ 7.0	76.2 $\pm$ 2.8	75.1 $\pm$ 5.1	52.3 $\pm$ 2.8
DIFFPOOL (Ying et al., 2018)	66.1 $\pm$ 7.7	78.8 $\pm$ 3.1	74.4 $\pm$ 4.0	50.1 $\pm$ 3.2
SORTPOOL (Zhang et al., 2018)	69.5 $\pm$ 6.3	79.2 $\pm$ 3.0	77.5 $\pm$ 2.7	53.1 $\pm$ 2.9
TOPKPOOL (Gao & Ji, 2019)	68.4 $\pm$ 6.4	79.1 $\pm$ 2.2	77.8 $\pm$ 5.1	53.7 $\pm$ 2.8
SAGPOOL (Lee et al., 2019)	69.0 $\pm$ 6.6	78.4 $\pm$ 3.1	77.8 $\pm$ 2.9	53.1 $\pm$ 2.8
ATTNPOOL (Girdhar & Ramanan, 2017)	71.2 $\pm$ 8.0	77.5 $\pm$ 3.3	77.1 $\pm$ 4.4	53.8 $\pm$ 2.5
COVPOOL (Wang et al., 2019a)	73.3 $\pm$ 5.1	80.1 $\pm$ 2.2	72.1 $\pm$ 5.1	47.8 $\pm$ 2.7
SOPOOL$_{attn}$ (Wang & Ji, 2020)	72.9 $\pm$ 6.2	79.4 $\pm$ 3.2	78.1 $\pm$ 4.0	54.3 $\pm$ 2.6
SOPOOL$_{bimap}$ (Wang & Ji, 2020)	75.0 $\pm$ 4.3	80.1 $\pm$ 2.7	78.4 $\pm$ 4.7	54.6 $\pm$ 3.6
Neural Pooling 1	74.5 $\pm$ **3.7**	**80.6** $\pm$ **2.7**	**79.0** $\pm$ **2.3**	**55.1** $\pm$ **2.2**
Neural Pooling 2	**76.2** $\pm$ **4.2**	79.6 $\pm$ 3.0	**78.5** $\pm$ **2.4**	54.4 $\pm$ **1.9**

Consider, $\boldsymbol{H}$ = [$\boldsymbol{h}_1$, $\boldsymbol{h}_2$, ..., $\boldsymbol{h}_n$]^T = GNN($\boldsymbol{A}$, $\boldsymbol{X}$) $\in \mathbb{R}^{n \times f}$ where rows of $\boldsymbol{H}$, $\boldsymbol{h}_i \in \mathbb{R}^f$, i = $1, 2, ..., n$, are representations of n nodes. Consider a direct application of second-order graph pooling to obtain the graph representation $\boldsymbol{h}_G$ as:

$$\boldsymbol{h}_G = \text{Flatten}(\boldsymbol{H}^T \boldsymbol{H}) \in \mathbb{R}^{f^2 \times 1} \tag{11}$$

where $\boldsymbol{H}^T$ denotes the transpose of $\boldsymbol{H}$.

However, it causes an explosion in the number of training parameters in the following classifier when f is large, making the learning process harder to converge and easier to overfit. While each layer in a GNN usually has outputs with a small number of hidden units (e.g. 16, 32, 64), it has been pointed out that graph representation learning benefits from using information from outputs of all layers, obtaining better performance and generalization ability. It is usually achieved by concatenating outputs across all layers in a GNN. In this case, $\boldsymbol{H}$ has a large final f, making direct use of second-order pooling infeasible. For example, if a GNN has 5 layers and each layer's outputs have 32 hidden units, f becomes 32×5 = 160. Suppose $\boldsymbol{h}_G$ is sent into a 1-layer fully-connected classifier for c graph categories in a graph classification task. It results in $160^2 c = 25,600c$ training parameters, which is excessive. We omit the bias term for simplicity. On the other hand, both of our proposed novel graph pooling methods significantly reduce the number of training parameters. In the case of Neural Pooling Method 1, considering the previous example if f' is chosen to be 64, and f is 160, then the total number of trainable parameters in the 2 $FCLs$ and a 1-layer fully-connected c class classifier will be $(160 \times 64) + 64 + 64c = 10,304 + 64c$ notably reducing the number of parameters as compared to $25,600c$. In the case of Neural Pooling Method 2, if f'' is chosen to be 64, f' as 32 and f is 160, then the total number of trainable parameters in the 2 $FCLs$ and a 1-layer fully-connected c class classifier will be $(160 \times 64) + (64 \times 32) + 322c = 12,288 + 1024c$ again reducing the number of parameters when compared to $25,600c$.

7 Conclusion

In this work, we propose to perform graph representation learning with Neural Pooling, by pointing out that Neural Pooling can naturally solve the challenges of graph pooling. Neural Pooling is more powerful than existing graph pooling methods, since it is capable of using all node information, collecting second-order statistics that encode feature correlations and topology information and leverage the ability of neural networks to learn from underlying data, making them more powerful. Our

proposed methods solve the practical problems incurred by directly using second-order pooling with GNNs. To demonstrate the effectiveness and superiority of our methods, we perform experiments on graph classification tasks in the bio-informatics and social network domains to demonstrate the effectiveness and superiority of our proposed methods. Experimental results show that our methods improve the performance significantly and consistently. An interesting future work direction could be to extend our methods for hierarchical graph pooling, where the output is a is a pseudo graph with fewer nodes than the input graph. It is used to build hierarchical GNNs, where hierarchical graph pooling is used several times between GNN layers to gradually decrease the number of nodes.

REFERENCES

KM Borgwardt, Stefan Schönauer, SVN Vishwanathan, Alexander Smola, and HP Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1:i47–56, 01 2005.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 111–118, Haifa, Israel, June 2010. Omnipress. URL http://www.icml2010.org/papers/638.pdf.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845, 2016.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2224–2232. Curran Associates, Inc., 2015.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, pp. 417–426, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488.

Hongchang Gao, Jian Pei, and Heng Huang. Conditional random field enhanced graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 276–284, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330888. URL https://doi.org/10.1145/3292500.3330888.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2083–2092. PMLR, 2019. URL http://proceedings.mlr.press/v97/gao19a.html.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html.

Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 34–45, 2017. URL http://papers.nips.cc/paper/6609-attentional-pooling-for-action-recognition.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1024–1034. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3734–3743. PMLR, 2019. URL http://proceedings.mlr.press/v97/lee19c.html.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4212–4221. PMLR, 2019a. URL http://proceedings.mlr.press/v97/ma19a.html.

Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. KDD '19, pp. 723–731, New York, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330982. URL https://doi.org/10.1145/3292500.3330982.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2014–2023. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/niepert16.html.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014. URL http://dl.acm.org/citation.cfm?id=2670313.

Hannu Toivonen, Ashwin Srinivasan, Ross D. King, Stefan Kramer, and Christoph Helma. Statistical evaluation of the Predictive Toxicology Challenge 2000–2001. *Bioinformatics*, 19(10): 1183–1193, 07 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg130. URL https://doi.org/10.1093/bioinformatics/btg130.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li. Deep cnns meet global covariance pooling: Better representation and generalization. *CoRR*, abs/1904.06836, 2019a. URL http://arxiv.org/abs/1904.06836.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, pp. 2022–2032, New York, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313562. URL https://doi.org/10.1145/3308558.3313562.

Z. Wang and S. Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5449–5458. PMLR, 2018. URL http://proceedings.mlr.press/v80/xu18c.html.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783417. URL https://doi.org/10.1145/2783258.2783417.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 4805–4815, 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. *AAAI*, 2018.