Article

Real Quadratic-Form-Based Graph Pooling for Graph Neural Networks

Youfa Liu 1 and Guo Chen 2,*

1 College of Informatics, Huazhong Agricultural University, Wuhan 430071, China; liuyoufa@mail.hzau.edu.cn
2 School of Business, Hubei University, Wuhan 430062, China
* Correspondence: guochen@hubu.edu.cn

Abstract: Graph neural networks (GNNs) have developed rapidly in recent years because they can work over non-Euclidean data and possess promising prediction power in many real-world applications. The graph classification problem is one of the central problems in graph neural networks, and aims to predict the label of a graph with the help of training graph neural networks over graph-structural datasets. The graph pooling scheme is an important part of graph neural networks for the graph classification objective. Previous works typically focus on using the graph pooling scheme in a linear manner. In this paper, we propose the real quadratic-form-based graph pooling framework for graph neural networks in graph classification. The quadratic form can capture a pairwise relationship, which brings a stronger expressive power than existing linear forms. Experiments on benchmarks verify the effectiveness of the proposed graph pooling scheme based on the quadratic form in graph classification tasks.

Keywords: graph neural networks; graph pooling; quadratic form

1. Introduction

Artificial intelligence has been prosperous for recent years due to the revolution of deep learning. Artificial neural networks make an attempt to simulate the power of the human brain, and approach the level of human beings in some tasks. At the beginning, deep neural networks simply dealt with regular data, such as text and images. However, irregular data occur frequently in the real world. Developing effective techniques towards this scenario is desirable. Later on, graph neural networks (GNNs) occurred gradually [1] and gave rise to wide research. This technique has many realistic applications, such as recommendation systems [2], traffic prediction [3] and drug target prediction [4].

Formally, GNNs are similar to deep neural networks in a layer-by-layer manner. They differ in that GNNs consider the relationship between nodes, which is reflected by edges in the underlying graph structure. As a result, GNNs not only use the features of data, but also consider the pairwise relation between data. From the statistical viewpoint, the data features in a graph are no longer independent. GNNs are one of the most popular techniques used to deal with this case.

GNNs learn low-dimensional embeddings for nodes in a graph by aggregation and transformation. The aggregator usually acts on a neighbourhood of a node. The size of the neighbourhood implies the range of dependence of a node with other nodes. Formally, the feature representation of a node in the l-th layer of a graph neural network is generated by an aggregation function acting on the representations from the (l-1)-th layer. The convolution operation in deep neural networks can be regarded as a special aggregation over grids in a district. Consecutive aggregation can lead to hierarchical node-level feature representations. The resulting representation can be used for different purposes, such as semi-supervised node classification and graph classification.

In this paper, we concern the graph classification problem. Facing this problem, we need to transform resulting embeddings into graph-level features so that a unified classifier would be able to make the final prediction.
can be followed for classification. This step is often called graph pooling [5], and produces graph-level features. In the training set, the size of a graph may be inconsistent; the graph pooling strategy must be adaptive for different sizes. In other words, the graph pooling strategy should be invariant to arbitrary sizes. In addition, it should be invariant to the order of nodes in a graph. Otherwise, an essentially same graph will produce multiple prediction results due to the change in the node’s order. This ambiguity should be avoided. For this reason, invariant GNNs have been intensively studied, such as [6,7]. Recall that GNNs are invariant if their output does not depend on the order of the nodes’ input. Recently, Keriven and Peyré [6] have provided universal approximation theorems for invariant and equivariant GNNs. The invariant property makes the output of graph neural networks stable and adaptive, which is effective in real world applications.

Previous works on graph neural networks for graph classification typically combine the graph pooling and multi-layer perceptron for generating final graph-level features [6]. This kind of strategy has two drawbacks: (1) this scheme may bring many parameters due to the introduction of multi-layer perceptron; (2) existing pooling schemes using a linear form neglect the possible pairwise relationship. Hence, we propose a real quadratic-form-based graph pooling framework to generate graph-level features without further multi-layer perceptron before performing classification. It comprises quadratic-form-based expressions. Generally, the number of quadratic forms should coincide with classes. By comparison, quadratic-form-based graph pooling layers need the least parameters and take the pairwise relationship into account. It is worth mentioning that the proposed quadratic-form-based graph pooling can be easily implemented by popular open-source deep learning frameworks, such as TensorFlow and PyTorch.

The reminder of this article is organised as follows. The second section provides some related works on graph neural networks for graph classification. The third section provides some preliminaries on problem formulation and the procedure of graph classification. The fourth section is about the proposed quadratic-form-based graph pooling frameworks. Experiments on benchmarks are conducted in the fifth section. Finally, we conclude this paper.

2. Related Works

In recent years, graph neural networks [1] have become a hot research topic in the field of machine learning. They can be used for semi-supervised node classification, link prediction and graph classification.

Graph convolution networks (GCNs) [1] are important and have many variants [2,3,6,7]. Their graph convolution is constructed by a localised first-order approximation of spectral graph convolution. Mathematically, it is approximated with the help of a truncated summation of Chebyshev polynomials.

Graph isomorphism networks (GIN) [8] are simple architectures whose expressive power can be comparable to the Weisfeiler–Lehman graph isomorphism test [9–11], which testifies whether two graphs are essentially identical in the sense of topology. It can fit the training data well in most cases.

WL [12] can rapidly extract features by the Weisfeiler–Lehman graph isomorphism test. It transforms a graph into a sequence of graphs, whose node attributes reveal topological and label information.

Graph pooling is a key step in the graph classification problem with graph neural networks. Second-order graph pooling (SOPOOL) [13] can treat the challenge of variable sizes and isomorphic structures of graphs. Bilinear mapping, attentional pooling and hierarchical pooling are also developed in [13].

No matter how the node’s order in a graph changes, GNNs must be invariant or equivariant (to permutation) because the relative graph structure is not changed essentially. There are few works on invariant and equivariant graph neural networks, such as [6,7]. Universal approximation theorems are provided for a better understanding of invariant
and equivariant networks in [6], which extends the classical universal approximation result for a multi-layer perceptron (MLP) with a single hidden layer [14].

In heterogeneous GNNs, aggregation is often realised by meta-paths. HPN [15] employs an appropriate weighting strategy such that deeper embeddings become distinguishable. HetGNN [16] samples heterogeneous neighbours of a fixed size and then uses two modules to aggregate this neighbourhood information. HHIN [17] uses the random walk to generate meta-paths and the hyperbolic distance to evaluate proximity.

3. Preliminaries

In this section, we formulate the problem setting and the procedure of graph classification. The problem setting is stated mathematically, which gives a clear aim for the research. The procedure of graph classification provides the routine of the general pipeline on how to realise this aim.

3.1. Problem Setup and Notations

A graph $G$ consists of vertex set $V$ and edge set $E$. We write $G = <V,E>$. For any node $v$ in $V$, it is assigned a feature $x_v \in \mathbb{R}^n$. Here, this feature reflects the quantitative information of node $v$. Let $A_G$ be the adjacency matrix whose entry is 1 or 0. If two nodes are connected, i.e., there is an edge between them, the value is 1. Otherwise, the value is 0. The adjacent can be naturally given; for example, a molecule. In addition, the adjacent can also be constructed by people. For example, two nodes are connected if the similarity between them is high. Let $X_G = [x_{G,1}, x_{G,2}, \ldots, x_{G,m}] \in \mathbb{R}^{n \times m}$ be the feature matrix of graph $G$.

Given a training set $\{<G_i, y_i>\}_{i=1}^N$ with graph labels, the task of graph classification is to establish a model on the basis of this dataset. The type of model in this paper is confined to graph neural networks. As a matter of fact, $G_i$ can be regarded as a sample from an unknown distribution.

3.2. Procedure of Graph Classification

The procedure of graph classification using graph neural networks contains the following five steps.

Step 1. Aggregate the information of the neighbourhood by some aggregation function. Mathematically, this is characterised as

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{x_u^{(k-1)} : u \in N(v)\right\}\right),$$

The common aggregation is SUM and AVG. Formally, the SUM operator-based aggregation is

$$a_v^{(k)} = \text{SUM}\left(\left\{x_u^{(k-1)} : u \in N(v)\right\}\right) = \sum_{u \in N(v)} x_u^{(k-1)}.$$  

The AVG operator-based aggregation is

$$a_v^{(k)} = \text{AVG}\left(\left\{x_u^{(k-1)} : u \in N(v)\right\}\right) = \frac{1}{|N(v)|} \sum_{u \in N(v)} x_u^{(k-1)},$$

where $N(v)$ is the neighbourhood of node $v$ and $|N(v)|$ denotes the number of set $N(v)$.

Step 2. Combine the aggregated features with the feature from the last layer. Mathematically, this is characterised as

$$x_v^{(k)} = \text{COMBINE}^{(k)}\left(\left\{x_u^{(k-1)}, a_v^{(k)}\right\}\right),$$

where $x_v^{(k-1)}$ is the feature of node $v$ from the last layer and $a_v^{(k)}$ is the aggregated feature of node $v$.
Step 3. Use a graph pooling scheme to obtain a graph-level feature.

Step 4 (Optional). Employ a multi-layer perceptron to obtain a final graph-level feature.

Step 5. Choose an appropriate classifier.

Step 1 and Step 2 can proceed alternatively. The meaning of aggregation in Step 1 is the integration of representations in the neighbourhood of a node. In Figure 1, we have illustrated the node \( v \) and its neighbourhood. By the above problem setting, each node admits a feature representation. These feature representations can be gathered in some way. Due to the fact that these nodes are connected to the node \( v \), the aggregation makes sense. After aggregation, the resulting vector from the neighbourhood can be associated with the feature representation of node \( v \). In this way, the representation of node \( v \) is enriched with the help of the information from the neighbourhood.

\[ \{ \psi ( \mathbf{x}_1 ) \psi ( \mathbf{x}_2 ) \cdots \psi ( \mathbf{x}_k ) \} \]

**Figure 1.** The node \( v \) and its neighbourhood.

In Step 3, the graph pooling scheme should satisfy the invariant property when the order of node changes. In Figure 2, we simply illustrate this process. After obtaining the feature by the graph pooling operation, the multi-layer perceptron can be followed for generating features of a higher level. Of course, this step is not necessary. It may bring additional parameters from the multi-layer perceptron, which increases the burden of network optimisation.

\[ \mathbf{G} \mathbf{a} \]

**Figure 2.** The illustration of graph pooling.

In Step 4, a classifier can be employed over final graph-level features for graph classification. For example, the classifier can be chosen as a softmax classifier that produces the probabilistic output.

4. Graph Pooling Framework

Graph pooling is a main step in the procedure of graph classification. In this section, we proposed a novel graph pooling framework based on a (real) quadratic form that can
capture a possible pairwise relationship. We also provide an instantiation with regard to this framework.

4.1. Review of General Real Quadratic Form from Linear Algebra

In linear algebra, the real quadratic form is generally expressed as

\[
M_Q(x) = x^T Q x,
\]

Specifically,

\[
M_Q(x) = \sum_{i,j} Q_{ij} x_i x_j,
\]

where \( Q = [Q_{ij}] \) is a real symmetric matrix. This quadratic meaning comes from the degree of each term in polynomials. The quadratic form is positive definite if \( M_Q(x) \geq 0 \) and \( M_Q(x) = 0 \) if and only if \( x = 0 \). It is positive semi-definite if \( M_Q(x) \geq 0 \).

We give a concrete example as follows.

**Example 1.**

\[
M_Q(x) = x_1^2 + x_2^2 + x_3^2 + x_1 x_2
\]

\[
= \begin{bmatrix} x_1, x_2, x_3 \end{bmatrix} \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},
\]

where the middle matrix is \( Q \), which is real symmetric.

4.2. The Proposed Approach

Observing the structure of a (real) quadratic form, every term contains the multiplication of two elements. In other words, this is a quadratic coupling method that includes a pairwise relationship. Let \( F_G^{(k)} = \text{GNN}(X_G, A_G) \in \mathbb{R}^{n_G \times f_G} \), be the embeddings in the \( k \)-th layer of some architecture of GNNs that are extracted by some graph neural network, where \( n_G \) is the number of vertex sets and \( f_G \) is the latent dimension.

The architecture of GNNs is chosen as GIN [8] in this paper. The update formula of the nodes’ representation is

\[
x_G^{(k)} = \text{MLP}^{(k)} \left( x_G^{(k-1)} + \sum_{u \in N(v)} x_u^{(k-1)} \right),
\]

where \( \text{MLP}^{(k)} \) is the \( k \)-layer perceptron and \( N(v) \) denotes the neighbourhood of node \( v \).

Here, the aggregation function is summation, which performs a summation operation for representation in the neighborhood of nodes.

Let \( P = \text{ReLU}(M_Q(x)) \),

where \( x \) is a free parameter and \( Q \) is specified as

\[
Q = (F_G^{(k)})^T F_G^{(k)},
\]

and

\[
\text{ReLU}(z) = \begin{cases} 
  z, & \text{if } z \geq 0; \\
  0, & \text{otherwise}.
\end{cases}
\]
In fact, $Q$ can be regarded as a similarity matrix as

$$Q = \begin{bmatrix} Q_{1,1} & \cdots & Q_{1,f_G} \\ \vdots & \ddots & \vdots \\ Q_{f_G,1} & \cdots & Q_{f_G,f_G} \end{bmatrix},$$

where the inner product similarity between node $i$ and $j$ is

$$Q_{ij} = \langle F_G^{(k)}(G_i), F_G^{(k)}(G_j) \rangle.$$

Hence,

$$MQ(x) = \sum_{ij} x_i x_j Q_{ij}$$

means the weighted summation of pairwise similarities in which the weight in each term is also a pairwise product.

When the quadratic form is positive semi-definite, we have

$$\text{ReLU}(MQ(x)) = MQ(x).$$

The above setting of $Q$ exactly makes this equality hold because the product of the transpose of a matrix with itself must be positive semi-definite.

It is easy to implement this kind of quadratic form in PyTorch by invoking “nn.Linear” and “torch.norm”.

The resulting representation is then

$$F_G^* = \begin{bmatrix} u_1^T Qu_1 \\ u_2^T Qu_2 \\ \vdots \\ u_C^T Qu_C \end{bmatrix},$$

where $C$ is the number of classes and $u_i$ is a parametric vector that serves as a connection. Here, we choose $C$ quadratic forms for generating a final graph-level feature.

When we use softmax classifier for this feature, the probabilistic output is

$$p_G = \text{softmax}(F_G^*),$$

where

$$\text{softmax}(F_G^*) = \begin{bmatrix} \exp(u_1^T Qu_1) / \sum_i \exp(u_i^T Qu_i) \\ \exp(u_2^T Qu_2) / \sum_i \exp(u_i^T Qu_i) \\ \vdots \\ \exp(u_C^T Qu_C) / \sum_i \exp(u_i^T Qu_i) \end{bmatrix}^T,$$

Then, the label of graph $G$ is predicted as

$$y_G \in \arg\max p_G.$$

We provide the information on the derivative of $P$ as follows, which is an ingredient in first-order optimisation.

Proposition 1. The partial derivatives of $P$ are

$$\frac{\partial P}{\partial x} = 2Qx.$$
and
\[
\frac{\partial P}{\partial F_G^{(k)}} = 2F_G^{(k)}xx^T.
\]

**Proof.** Recall that the expression of \( P \) is actually.

\[
P = x^T Q x.
\]

Through basic knowledge of calculus, we have
\[
\frac{\partial P}{\partial x} = 2Q x
\]

To derive the second partial derivative, we need the following transformations:
\[
P = x^T Q x = x^T (F_G^{(k)})^T F_G^{(k)} x = \text{tr}(F_G^{(k)} xx^T (F_G^{(k)})^T),
\]

where \( \text{tr}(\cdot) \) denotes the trace of matrix.

Then, we have
\[
\frac{\partial P}{\partial F_G^{(k)}} = \frac{\partial \text{tr}(F_G^{(k)} xx^T (F_G^{(k)})^T)}{\partial F_G^{(k)}} = 2F_G^{(k)} xx^T.
\]

4.3. Comparison

Recall that Step 4 is usually optional in the aforementioned procedure of the graph classification. That is, a multi-layer perceptron is employed to obtain final graph-level feature. Existing works on graph pooling typically add the multi-layer perceptron for a better prediction performance. The AVG and SUM are two simple graph pooling operators. The former is average and the latter is summation. The advantage of these two operators is that they are parameter-free. When a fully connected (FC) layer is followed, the amount of parameters is \( f_{GC} \). Hence, both AVG+FC and SUM+FC contain the \( f_{GC} \) parameters. The recent second-order pooling methods with attention SOPOOL\textsubscript{attn} [13] possess a subsequent multi-layer perceptron. As a matter of fact, it uses the one-layer perceptron, i.e., the fully connected layer. Hence, the amount of parameters is the sum of \( f_G \) and \( f_{GC} \), where \( f_G \) is the number of parameters in attention and \( f_{GC} \) is the number of parameters in the fully connected layer. As a summarisation, we display the comparison result in Table 1.

| Schemes            | Multi-Layer Perceptron | #Parameters          |
|--------------------|------------------------|----------------------|
| AVG + FC           | Yes                    | \( f_{GC} \)         |
| SUM + FC           | Yes                    | \( f_{GC} \)         |
| SOPOOL\textsubscript{attn} + FC | Yes                  | \( f_G(1 + C) \)     |
| Ours               | No                     | \( f_{GC} \)         |

5. Experiments
5.1. Dataset Description

There were nine graph classification datasets from [18] in our experiments. They can be roughly categorised into two kinds: bioinformatics and social network datasets. For clarity, we introduce them one by one. The statistics of the datasets is displayed in Table 2.
Table 2. Statistics of datasets.

| Datasets  | #Graphs | #Classes | # Nodes (Max) | #Nodes (Avg) |
|-----------|---------|----------|---------------|--------------|
| MUTAG     | 188     | 2        | 28            | 18           |
| PTC       | 344     | 2        | 109           | 25.6         |
| PROTEINS  | 1113    | 2        | 620           | 39.1         |
| NCI1      | 4110    | 2        | 111           | 29.9         |
| COLLAB    | 5000    | 3        | 492           | 74.5         |
| IMDB-B    | 1000    | 1        | 136           | 19.8         |
| IMDB-M    | 1500    | 3        | 89            | 13.0         |
| RDT-B     | 2000    | 2        | 3783          | 429.6        |
| RDT-M5K   | 5000    | 5        | 3783          | 508.5        |

MUTAG is a bioinformatics dataset. It contains 188 graphs and each graph represents nitro compounds. The maximum nodes are 28 and average nodes are 18 in this dataset. Every node in a graph has one of seven discrete node labels [19]. There are two kinds of graph labels.

PTC is a bioinformatics dataset. It consists of 344 graphs. Every graph represents chemical compounds [20]. The maximum nodes are 109 and average nodes are 25.6 in this dataset. Every node bears one of 19 discrete node labels. There are two kinds of graph labels.

PROTEINS is a bioinformatics dataset. It comprises 1113 graph structures of proteins. Nodes in the graphs represent secondary structure elements and are assigned with discrete node labels. The maximum nodes are 620 and average nodes are 39.1 in this dataset. Edges mean that two nodes are connected along the amino-acid sequence in space. There are two kinds of graph labels.

NCI1 is a bioinformatics dataset. It includes 4110 graphs in all. Each graph represents chemical compounds [21]. The maximum nodes are 111 and average nodes are 29.9 in this dataset. Every node possesses one of 37 discrete node labels. There are two kinds of graph labels.

COLLAB is a scientific collaboration dataset. It contains 5000 graphs in total. Each graph is generated by ego-networks as [22]. The dataset originates from 3259 public collaboration datasets [23]. Every ego-network includes researchers from different fields, and its label is named by the field. The maximum nodes are 492 and average nodes are 74.5 in this dataset. There are three kinds of graph labels.

IMDB-BINARY is a movie collaboration dataset. It has 1000 graphs. Each graph corresponds to ego-networks for actors/actresses. The dataset is induced by collaboration graphs on Action and Romance genres. Actors/actresses are regarded as nodes and edges reveal that they collaborate with the same movie. The maximum nodes are 136 and average nodes are 19.8 in this dataset. The graph label is marked with the corresponding genre and the task is to predict the genre for graphs.

IMDB-MULTI is a multi-class version of IMDBBINARY. It has 1500 ego-networks and three extra genres, including Comedy, Romance and Sci-Fi. The maximum nodes are 89 and average nodes are 13.0 in this dataset.

REDDIT-BINARY is a dataset that includes 2000 graphs. Each graph corresponds to an online discussion thread. Nodes in a graph correspond to users that lie in the corresponding discussion thread, and an edge signifies that one user responded to another. The graph class number is 2. The maximum nodes are 3783 and average nodes are 429.6 in this dataset.

REDDIT-MULTI5K resembles REDDITBINARY. In total, there are 5000 graphs. REDDIT-MULTI5K are crawled data from five different subreddits that contain worldnews, videos, AdviceAnimals, aww and mildly interesting. The number of graph classes is five. The maximum nodes are 3783 and average nodes are 508.5 in this dataset.

5.2. Comparison Methods

The following methods were chosen as comparison.
WL [12]: On the basis of the Weisfeiler–Lehman test of isomorphism on graphs, it generates a graph sequence whose topological and label information are fused into attributes for future graph classification.

PATCHSCAN [24]: By mimicking image-based convolution that operates on locally connected regions, it extracts locally connected regions from graphs. It acts in a patch-like way.

DGCNN [25]: This is a kind of neural network that takes the underlying graph as an input and trains a classifier for graph classification.

GIN-0+AVG/SUM: This is a composite method that combines GINs [8] with an average operator or summation operator.

GIN-0+SOPOOL\textsuperscript{attn} [13]: Similarly, this is also a composite method that combines GINs [8] with pooling method SOPOOL\textsuperscript{attn} in [13].

5.3. Implementation Details

Quadratic-form-based graph pooling was inserted into flat GNNs from recent graph isomorphism networks (GINs) [8] that have strong expressive power. The GINs utilise AVG/SUM graph pooling to generate a graph-level feature. Specifically, SUM graph pooling was used on bioinformatics datasets and AVG graph pooling was chosen for social network datasets. Here, we employed the proposed quadratic-form-based graph pooling scheme and kept the other architecture of GINs unchanged. For the flat GNNs, we followed the same training process in [8]. All GINs used in experiments had five layers and two-layer perceptron with batch normalisation [26]. Adam optimiser [27] was chosen for optimisation of graph neural networks with annealing strategy, whose learning rate was initialised as 0.01 and decay rate was 0.5 by 50 epochs. Hidden dimension was tuned over [16, 32, 64, 128] and batch size was chosen from [32, 64, 128]. Ten-fold cross-validation was used for obtaining final experimental results. The best number of epochs was determined by the best cross-validation result.

For WL [12], the height parameter was set as 2. The classifier was chosen as support vector machine implemented by LIB-SVM [28]. For PATCHSCAN [24], we performed 1-dimensional WL normalisation. The width parameter was set as the average number of nodes. Receptive field sizes were tuned from [5, 10]. For DGCNN [25], AdaGrad algorithms [29] were used as optimiser, with learning rate 0.05. All weights were initialised by sampling from Gaussian distribution \( N(0, 0.01) \).

5.4. Experimental Result and Analysis

The experiments were performed over the aforementioned nine datasets. The experimental results are reported in Table 3. The last column is the result of our method, i.e., GIN-0 with the proposed quadratic-form-based graph pooling scheme. It can been seen from the experimental results that our method almost achieves the best results. It behaves better than the linear form, such as AVG, SUM and SOPOOL\textsuperscript{attn} [13]. The cause may be that the quadratic form is involved with the pairwise relationship. The quadratic form is a non-linear expression. It can extract non-linear representations under a pairwise pattern for nodes, while the linear form cannot obtain any pairwise information. The expression of the quadratic form includes second-order information because it belongs to a kind of second-order polynomial. We also found that the proposed method behaves better on bioinformatics datasets than social network datasets. We provide the numerical evidence for this in Table 4. The cause of this phenomenon may be that the objectivity of the graph structure in bioinformatics datasets is stronger than that in social network datasets, in which, the edge in a graph is not necessarily constructed by people. From Table 4, we can see that the metric of the standard deviation is close for two kinds of datasets. It may reveal that the proposed graph pooling scheme is relatively stable across domains.
Table 3. Graph classification accuracy (%). The best result is marked with boldface.

| Datasets | WL  | PATCHSCAN | DGCNN | GIN-0+ AVG/SUM | GIN-0+ SOPOOL | Ours  |
|----------|-----|------------|-------|----------------|----------------|-------|
| MUTAG    | 90.4 ± 5.7 | 92.6 ± 4.2  | 85.8 ± 1.7 | 89.4 ± 5.6  | 93.6 ± 4.1 | 94.2 ± 2.0 |
| PTC      | 59.9 ± 4.3  | 60.0 ± 4.8  | 58.6 ± 2.5 | 64.6 ± 7.0  | 72.9 ± 6.2 | 75.1 ± 3.1 |
| PROTEINS | 75.0 ± 3.1  | 75.9 ± 2.8  | 75.5 ± 1.0 | 76.2 ± 2.8  | 79.4 ± 3.2 | 80.5 ± 1.8 |
| NCHI     | 86.0 ± 1.8  | 78.6 ± 1.9  | 74.4 ± 0.5 | 82.7 ± 1.7  | 82.8 ± 1.4 | 84.7 ± 1.1 |
| COLLAB   | 78.9 ± 1.9  | 72.6 ± 2.2  | 73.8 ± 0.5 | 80.2 ± 1.9  | 81.1 ± 1.8 | 82.0 ± 1.5 |
| IMDB-B   | 73.8 ± 3.9  | 71.0 ± 2.2  | 70.0 ± 0.9 | 75.1 ± 5.1  | 78.1 ± 4.0 | 78.6 ± 2.5 |
| IMDB-M   | 50.9 ± 3.8  | 45.2 ± 2.8  | 47.8 ± 0.9 | 52.3 ± 2.8  | 54.3 ± 2.6 | 55.0 ± 2.4 |
| RDT-B    | 81.0 ± 3.1  | 86.3 ± 1.6  | 76.0 ± 1.7 | 92.4 ± 2.5  | 91.7 ± 2.7 | 91.8 ± 1.4 |
| RDT-M5K  | 52.5 ± 2.1  | 49.1 ± 0.7  | 48.7 ± 4.5 | 57.5 ± 1.5  | 58.3 ± 1.4 | 58.6 ± 1.4 |

Table 4. Comparison on average performance on two classes of datasets.

| Dataset Type          | Average Accuracy (%) | Standard Deviation |
|-----------------------|----------------------|--------------------|
| Boinformatics         | 83.63                | 2.0                |
| Social networks       | 73.20                | 1.8                |

6. Conclusions

In this paper, we consider the graph classification problem with graph neural networks. The quadratic-form-based graph pooling scheme is proposed. Under this scheme, the multi-layer perceptron is not necessarily followed and spends the smallest amount of parameters when compared with existing methods. It can be easily implemented in a popular deep learning framework, such as TensorFlow and PyTorch. Experiments demonstrate the effectiveness of the proposed graph pooling scheme, which is based on the quadratic form.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L. and G.C.; formal analysis, G.C.; investigation, Y.L.; resources, Y.L.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, G.C.; visualization, Y.L.; supervision, G.C.; project administration, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Natural Science Foundation of Hubei Province under grant number 2021CFB139 and the APC was funded by Project 2662020XXQD002 supported by the Fundamental Research Funds for the Central Universities.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. arXiv 2016, arXiv:1609.02907.
2. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; Yin, D. Graph Neural Networks for Social Recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13 May 2019.
3. Chen, C.; Li, K.; Teo, S.G.; Zou, X.; Wang, K.; Wang, J.; Zeng, Z. Gated residual recurrent graph neural networks for traffic prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
4. Lim, J.; Ryu, S.; Park, K.; Choe, Y.J.; Ham, J.; Kim, W.Y. Predicting drug-target interaction using a novel graph neural network with 3D structure-embedded graph representation. J. Chem. Inf. Mode. 2019, 59, 3981–3988. [CrossRef] [PubMed]
5. Gao, H.; Liu, Y.; Ji, S. Topology-aware graph pooling networks. IEEE Trans. Pattern Anal. Mach. Intell. 2021, 43, 4512–4518. [CrossRef] [PubMed]
6. Keriven, N.; Peyré, G. Universal invariant and equivariant graph neural networks. Adv. Neural Inf. Process. Syst. 2019, 32, 7092–7101.
7. Maron, H.; Ben-Hamu, H.; Serviansky, H.; Lipman, Y. Provably powerful graph networks. Adv. Neural Inf. Process. Syst. 2019, 32, 2156–2167.
8. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful are Graph Neural Networks? In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
9. San Martino, G.D.; Navarin, N.; Sperduti, A. Graph Kernels Exploiting Weisfeiler-Lehman Graph Isomorphism test extensions. In Proceedings of the International Conference on Neural Information Processing, Montreal, QC, Canada, 8–13 December 2014.

10. De Vries, G.K.D. A Fast Approximation of the Weisfeiler-Lehman Graph Kernel for RDF Data. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Prague, Czech Republic, 23–27 September 2013.

11. Huang, N.T.; Villar, S. A Short Tutorial on The Weisfeiler-Lehman Test and Its Variants. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Virtual Conference, 6–12 June 2021.

12. Shervashidze, N.; Schweitzer, P.; Leeuwen, E.J.V.; Mehlhorn, K.; Borgwardt, K.M. Weisfeiler-Lehman graph kernels. J. Mach. Learn. Res. 2011, 12, 2539–2561.

13. Wang, Z.; Ji, S. Second-order pooling for graph neural networks. IEEE Trans. Pattern Anal. Mach. Intell. 2020. [CrossRef]

14. Cybenko, G. Approximation by superpositions of a sigmoidal function. Mathematics Control. Signals Syst. 1989, 2, 303–314. [CrossRef]

15. Ji, H.; Wang, X.; Shi, C.; Wang, B.; Yu, P. Heterogeneous graph propagation network. IEEE Trans. Knowl. Data Eng. 2021. [CrossRef]

16. Zhang, C.; Song, D.; Huang, C.; Swami, A.; Chawla, N.V. Heterogeneous Graph Neural Network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019.

17. Wang, X.; Zhang, Y.; Shi, C. Hyperbolic Heterogeneous information network embedding. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.

18. Yanardag, P.; Vishwanathan, S.V.N. Deep Graph Kernels. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13 August 2015.

19. Debnath, A.K.; Lopez de Compadre, R.L.; Debnath, G.; Shusterman, A.J.; Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. J. Med. Chem. 1991, 34, 786–797. [CrossRef] [PubMed]

20. Toivonen, H.; Srinivasan, A.; King, R.D.; Kramer, S.; Helma, C. Statistical evaluation of the predictive toxicology challenge 2000–2001. Bioinformatics 2003, 19, 1183–1193. [CrossRef] [PubMed]

21. Wale, N.; Watson, I.A.; Karypis, G. Comparison of descriptor spaces for chemical compound retrieval and classification. Bioinformatics 2008, 14, 347–375.

22. Shrivastava, A.; Li, P. A New Space for Comparing Graphs. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Beijing, China, 17–20 August 2014.

23. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August 2005.

24. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning convolutional neural networks for graphs. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.

25. Atwood, J.; Towsley, D. Diffusion-convolutional neural networks. Adv. Neural Inf. Process. Syst. 2016, 29, 1993–2001.

26. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015.

27. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representation, San Diego, CA, USA, 7–9 May 2015.

28. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2011, 2, 1–27. [CrossRef]

29. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. 2011, 12, 2121–2159.