RESEARCH ARTICLE

A deep graph convolutional neural network architecture for graph classification

Yuchen Zhou, Hongtao Huo, Zhiwen Hou, Fanliang Bu*

School of Information Network Security, People's Public Security University of China, Beijing, China

* bufanliang@sina.com

Abstract

Graph Convolutional Networks (GCNs) are powerful deep learning methods for non-Euclidean structure data and achieve impressive performance in many fields. But most of the state-of-the-art GCN models are shallow structures with depths of no more than 3 to 4 layers, which greatly limits the ability of GCN models to extract high-level features of nodes. There are two main reasons for this: 1) Overlaying too many graph convolution layers will lead to the problem of over-smoothing. 2) Graph convolution is a kind of localized filter, which is easily affected by local properties. To solve the above problems, we first propose a novel general framework for graph neural networks called Non-local Message Passing (NLMP). Under this framework, very deep graph convolutional networks can be flexibly designed, and the over-smoothing phenomenon can be suppressed very effectively. Second, we propose a new spatial graph convolution layer to extract node multiscale high-level node features. Finally, we design an end-to-end Deep Graph Convolutional Neural Network II (DGCNNII) model for graph classification task, which is up to 32 layers deep. And the effectiveness of our proposed method is demonstrated by quantifying the graph smoothness of each layer and ablation studies. Experiments on benchmark graph classification datasets show that DGCNNII outperforms a large number of shallow graph neural network baseline methods.

OPEN ACCESS

Citation: Zhou Y, Huo H, Hou Z, Bu F (2023) A deep graph convolutional neural network architecture for graph classification. PLoS ONE 18(3): e0279604. https://doi.org/10.1371/journal.pone.0279604

Editor: Abel C.H. Chen, Chunghwa Telecom Co. Ltd., TAIWAN

Received: July 11, 2022

Accepted: December 10, 2022

Published: March 10, 2023

Data Availability Statement: The data underlying the results presented in the study are available from (https://github.com/muhanzhang/pytorch_DGCNN/tree/master/data).

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

1 Introduction

In recent years, the rapid development of Convolution Neural Networks (CNNs) [1] have achieved impressive results in many fields. CNNs are suitable for processing Euclidean structure data with translation invariance such as image, text, speech, etc. This kind of data takes one of the data units as the central node and has the same number of neighbors, so the features of the data can be extracted sequentially by defining a globally shared convolution kernel. However, in the real world, the non-Euclidean structure data of knowledge graph, social network, chemical molecule and other graph structures are increasing, which is characterized by non-translation invariance, and the number of neighboring nodes may be different, so the convolution kernel cannot be used to sequentially extract the data information of the same structure. The application of CNNs on non-Euclidean structure data has limitations, so the Graph Convolutional Networks (GCNs) [2] were proposed to generalize CNNs to the above

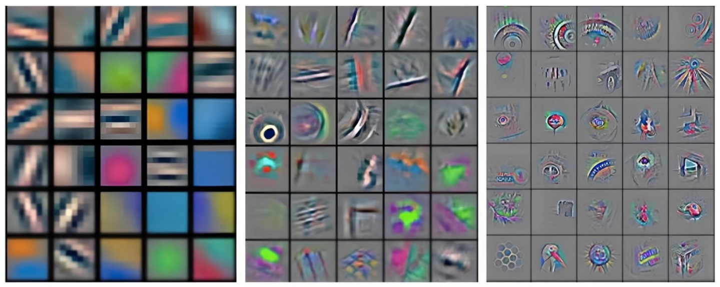

Fig 1. Object features extracted by neural networks with different depths [13].

https://doi.org/10.1371/journal.pone.0279604.g001

graph-structured data. GCNs and its variants are also widely used in many fields, such as traffic prediction [3], computer vision [4–6], machine translation [7, 8], disease prediction [9, 10], social analysis [11], recommendation system [12] and so on.

In the field of image recognition, theoretically, the deeper the network, the more abstract high-level information can be extracted. The multi-layer network structure of the convolutional neural networks can learn the features of different levels of the image, and the extracted features are also richer. The shallow network has a small perception area and can learn the local regional features of the image, such as the edge and color of the object. The deep network has a larger perception area and can learn more abstract features of images, such as shape, contour, property, location and other high-dimensional features of objects. The ability of deep convolutional neural networks currently used in image recognition tasks to recognize objects even surpasses humans. As shown in Fig 1, the shallow network can only extract low-level features such as colors and lines, the intermediate network can extract middle-level features such as edges and contours, and the deep network can extract abstract high-level features such as shape, property, and category.

Therefore, we hope to introduce the concept of deep network in CNNs into GCNs, so that GCNs can also achieve the effect of deep CNNs, which can be used to extract high-order abstract features of nodes. However, most current state-of-the-art GCN models are no more than 3 to 4 layers deep and are usually limited to very shallow models. Q. Li et al. [14] pointed out that the shallow GCN model such as double-layer GCN [2] has its limitations and requires a large number of extra tags for model training. When only a few tags are given, the shallow GCN model is difficult to aggregate multi-hop neighborhoods information to the central node and is easily affected by the local properties of the convolution filter. Some papers [14–16] have analyzed the limitations of GCN, and pointed out that overlaying too many graph convolutional layers will lead to the over-smoothing problem, so that the features of nodes tend to be consistent, and different nodes will become indistinguishable. Xu et al. [17] studied the expressive ability of popular GNN variants and found that if GNN has strong expressive ability, different multiple aggregates must be synthesized into different representations. They also propose a theoretical framework to analyze the expressive ability of GNN to capture different graph structures, and show that popular GNN variants cannot learn to distinguish some simple graph structures. Alon et al. [18] pointed out that one of the main problems of training GNN is that it is difficult for them to propagate information between remote nodes in the graph. Each node in the graph has a very large number of K-order neighbors that when a remote node information is transmitted, it will be compressed / distorted. This phenomenon is called "over-squashing". Similar problems also appear in CNNs. Due to the chain rule, with the deepening of the network, the gradient may vanish or explode in the process of back propagation, and the deep neural network may degenerate into a shallow neural network, or even the network becomes unlearnable. To solve the problem of gradient vanishing / explosion in CNNs, He et al. [19] proposed the ResNet to make skip connections in some layers of the network, weaken the strong connection between each layer by adding residual connections and nonlinear transformation to achieve identity mapping, and get better results as the network deepens. On the basis of ResNet, Huang et al. [20] proposed DenseNet, which connects all layers to each other, realizes feature reuse, and makes it easier to achieve gradient back propagation and make the network easier to train.

In this paper, we hope to improve the performance of graph convolution network by deepening it, extract richer high-order abstract features of nodes by using deep network, and verify it on graph classification task. We use the ideas of ResNet and DenseNet in convolutional neural networks to conduct research on deep graph convolutional networks. In order to ensure that the deep network can at least achieve the performance of the shallow network, we transfer the shallow node features to the deep network, and reuse the features before each layer of the network. At the same time, the identity mapping mechanism is introduced into the linear transformation of node features. We show how to successfully integrate the concepts and methods of deep convolution neural network into a new general framework of graph neural network, and extensively analyze the accuracy and stability of the proposed deep graph convolutional network based on the new general framework. We show that GCN up to 32 layers can be successfully trained by using a non-local message passing framework, and we intend to test deeper networks in future research. Compared with 22 baseline methods such as 6 graph kernels and 16 other graph neural network methods, the proposed deep GCN model DGCNNII achieves first place in graph classification task on 4/5 bioinformatics datasets. Compared with DGCNN [21], when only the graph convolution part of the model is replaced, the accuracy of DGCNNII on the 5 datasets is about 3.96%–17.88% higher than that of DGCNN.

Our contributions in this paper are as follows. 1) We propose a novel general framework for graph neural networks called **Non-local Message Passing (NLMP)**. 2) Based on the NLMP framework, we propose a new spatial graph convolution layer to extract node multiscale high-order node features. 3) A novel end-to-end deep learning model DGCNNII for graph classification tasks is proposed. It can directly accept graphs as input without any pre-processing of the data. 4) Experimental results on benchmark graph classification datasets show that our **Deep Graph Convolutional Neural Network II (DGCNNII)** significantly outperforms graph kernel methods and numerous other deep learning methods on graph classification task.

2 Related works

2.1 Message Passing Neural Network (MPNN)

In recent years, the general framework of Message Passing Neural Network (MPNN) proposed by Gilmer et al. [22] has been applied to various graph related tasks, especially in graph supervised learning tasks. Such as SafeDrug, a drug recommendation model based on MPNN framework proposed by Yang et al. [23] which can encode the connectivity and function of drug molecules to achieve safe and accurate drug recommendation. Dasoulas et al. [24] proposed a graph neural network called Colored Local Iterative Procedure (CLIP) on the basis of MPNN, which used color to eliminate the ambiguity of the same node, and proved that this representation is a universal approximator of continuous functions on graphs with node properties, and demonstrate that this simple coloring scheme can theoretically and empirically improve the expressiveness of message passing neural networks. In terms of node update, the general framework of MPNN is mainly divided into two parts, one is to aggregate the information of neighboring nodes and edges around the vertex, and the other is to update the node information iteratively by fusing the vertex features of the previous iteration with the vertex features of the current aggregation. Specifically, see the following equations:

$$m_i^{(t+1)} = \sum_{v_j \in N(v_i)} M_t\left(h_i^{(t)}, h_j^{(t)}, e_{ij}\right) \tag{1}$$

$$h_i^{(t+1)} = U_t\left(h_i^{(t)}, m_i^{(t+1)}\right) \tag{2}$$

M and U in the above Eqs (1) and (2) represent the message transfer function and the message update function, respectively. where v_i denotes vertex i, $N(v_i)$ denotes the first-order neighbor nodes of vertex i, h_i denotes the feature vector of vertex i, e_{ij} denotes the edge $<v_i, v_j>$, m_i denotes the feature vector of vertex i after aggregation operation, and t denotes the rounds of message propagation iterations.

Eq (1) represents the message passing aggregation operation, vertex i aggregates its feature vector $h_i^{(t)}$ at moment t with its first-order neighbor nodes feature vector $h_{N(v_i)}^{(t)}$ and the information of edges to obtain the node feature vector $m_i^{(t+1)}$ after aggregate operation. Eq (2) represents the node information update operation. The node feature vector $h_i^{(t)}$ at the previous moment and the node feature vector $m_i^{(t+1)}$ obtained by this iteration are aggregated through the update function U to update the node information.

The core of the MPNN framework is the message passing function and the message update function. Different improvements are made for these two parts. In principle, any graph neural network model can be designed based on this framework. From the perspective of message passing framework, many variants of graph neural network models have been proposed, and these models have achieved excellent results in node classification and graph classification tasks. Table 1 lists the message passing functions and message update functions of some classical graph neural network models.

2.2 Non-local Neural Network (NLNN)

X. Wang et al. [28] proposed a Non-local Neural Network (NLNN) model applied in the field of computer vision, which uses deep neural networks to capture remote dependencies. The idea of non-local operation is based on the generalization of non-local mean operation proposed by Buades et al. [29] and others. Its core idea is to perform weighted calculation on the features of all specific locations. The specific location can be the pixel coordinates of the spatial dimension or the time coordinate of the time dimension. When it is migrated to the graph structure data, the specific location can be replaced by a node. The deep graph convolution

Table 1. Different graph neural network models based on MPNN framework.

Model	Message Passing Function	Message Update Function
GCN [2]	$m^{(t+1)} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(t)}W^{(t)}$	$H^{(t+1)} = ReLU(m^{(t+1)})$
GraphSAGE-mean [25]	$m^{(t+1)} = W^{(t)} \cdot MEAN(\{h_i^{(t)}\} \cup \{h_j^{(t)}, v_j \in N(v_i)\})$	$h_i^{(t+1)} = ReLU(m^{(t+1)})$
R-GCN [26]	$m^{(t+1)} = \sum_{r \in R} \sum_{j \in N_i^{(r)}} \frac{1}{c_{i,r}} W_r^{(t)} h_j^{(t)} + W_0^{(t)} h_i^{(t)}$	$h_i^{(t+1)} = ReLU(m^{(t+1)})$
DGCNN [21]	$m^{(t+1)} = \tilde{D}^{-1}\tilde{A}H^{(t)}W^{(t)}$	$H^{(t+1)} = tanh(m^{(t+1)})$
DiffPool [27]	$m^{(t+1)} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(t)}W^{(t)}$	$H^{(t+1)} = ReLU(m^{(t+1)})$
GIN [17]	$m^{(t+1)} = (1 + \epsilon^{(t+1)}) \cdot h_i^{(t)} + \sum_{v_j \in N(v_i)} h_j^{(t)}$	$h_i^{(t+1)} = MLP^{(t+1)}(m^{(t+1)})$

https://doi.org/10.1371/journal.pone.0279604.t001

proposed in this paper can extend the specific position to multiple-hop neighborhood nodes after several iterations. For large-scale graphs, as long as the graph convolution layers are deep enough, the dependency of each node can be extended to the whole graph for capturing more comprehensive structural information of nodes, but need to solve the over-smoothing problem caused by multi-layer graph convolution, the solution will be proposed below.

In the paper of Buades et al. [29], given discrete noise image $v = \{v(i)|i \in I\}$, where $v(i)$ represents the pixel value of pixel i and I represents all the pixels of the image. For pixel i, the estimated value $NL[v](i)$ represents the weighted average of all pixels in the image, and the specific equation is as follows:

$$NL[v](i) = \sum_{j \in I} w(i,j)v(j) \tag{3}$$

Where $\{w(i,j)|j \in I\}$ represents the similarity weight between pixel i and pixel j, and satisfies $0 \le w(i,j) \le 1, \sum_j w(i,j) = 1$. The estimated value of pixel i is represented by calculating the sum of products with similarity weight between pixel i and pixel $j(j \in I)$. The similarity weight in this method is similar to the attention weight in the "Attention Mechanism". Therefore, the NLNN framework can be regarded as the unification of various current self-attention-based methods [30–32]. In the field of graph convolution, analogous to the definition of Gaussian filtering Eq (4), the generalized node non-local feature aggregation operation is defined as Eq (5):

$$\mathrm{GB}[I]_p = \sum_{q \in S} G_\sigma(||p-q||)I_q \tag{4}$$

$$h_i^{(t+1)} = \frac{1}{C(h)} \sum_{\forall j} f\left(h_i^{(t)}, h_j^{(t)}\right) \cdot g\left(h_j^{(t)}\right) \tag{5}$$

In the above Eq (4), $\mathrm{GB}[I]_p$ represents the value of the pixel p after Gaussian blurring, I_q represents the value of pixel q, S represents the non-local adjacent region of pixel p, and G_σ represents Gaussian function, which is used to calculate the weight value. $h_i^{(t)}$ in the above Eq (5) represents the feature of the target node i at time t, $h_j^{(t)}$ represents the feature of all nodes j related to the target node i at time t, and $f(h_i^{(t)}, h_j^{(t)})$ is used to calculate the attention coefficient between node i and node j, $1/C(h)$ is used to normalize the results, and $g(h_j^{(t)})$ represents the function that transforms the features of the input node j. The idea of NLNN framework is to aggregate all the node features related to the target node according to different attention weights to achieve the update of the target node features. When designing the model for different problems, function f and function g can be designed differently. The specific design of the model in this paper will be introduced in Section 3.

2.3 Graph classification

The framework and model proposed in this paper will be applied to the task of graph classification. Graph classification is also called graph property prediction, i.e., given a set of graphs, the goal is to learn the mapping relationship between the graph and the corresponding category label, and apply the trained model and parameters to the category prediction of unknown graphs. For example, in the field of chemical molecules, the structural of molecules can be seen as graph structure data, and the class prediction of molecular structure is used to determine the mutagenicity, toxicity, and anticancer activity of compound molecules [33, 34]. In molecular biology, by classifying and predicting the protein structure, we can judge whether the unknown protein is an enzyme, so as to determine whether the unknown protein has a therapeutic effect on a disease [35, 36]. The methods of graph classification mainly include methods based on graph kernels [37, 38], methods of graph similarity matching [39] and methods based on deep learning.

At present, the common steps for predicting graph properties based on deep learning are: 1) Firstly, the existing graph convolution method [2, 17, 21, 25–27] is used to extract node features, and the embedded representation of nodes is obtained. 2) Then aggregate the embedded representation of all the nodes in the graph to represent the graph embedded representation. 3) Finally, graph embedding representation is used to predict graph properties. For the first step, we can use one of the existing GCN methods to learn the node embedding representation. In the second step, we can use the readout function (pooling function) to read out the nodes in a certain order, and embed the nodes in a specific order as the graph embedding representation. In the last step, we classify the graph according to the embedding of each subgraph. Therefore, the research on graph property prediction using deep learning mainly focuses on the following two aspects. One is how to extract node features more accurately. Node representation learning is the precondition of graph representation learning. In recent years, a lot of work has been done on how to learn node embedding or graph embedding through various graph neural networks [40]. The second is how to design a readout function to perform a pooling operation on the graph. For isomorphic graphs, the readout function should be read out in a consistent order according to the role of nodes in the graph to ensure that the structural features of the graph do not change after the pooling operation.

For node representation learning, the node representation learning methods used in graph classification models such as EigenGCN [41], DGCNN [21], DiffPool [27], SAGPool [42] all adopt message passing GCN and its variants. For the graph pooling readout function, the feature representation of all nodes can be simply added or averaged as the feature representation of the graph, but this method will ignore many key nodes and structural information in the graph. Therefore, many more complex node readout methods have been proposed in recent years. For example, the pooling operation of the EigenGCN model proposed by [41] is divided into two parts: First, the large graph is divided into multiple subgraphs, each sub-graph is pooled to get a super node, and multiple super nodes form a coarsening graph. The original graph signal is then converted into a graph signal defined on the coarsening graph using EigenPooling. The Graph U-Nets model with an encoder-decoder structure proposed by Gao & Ji [43] can perform graph pooling and unpooling operations on graphs.

2.4 Over-smoothing in node-property prediction

In the field of image recognition, the deepening of the network will lead to gradient vanishing or gradient explosion, and the problem of network degradation occurs. In the field of graph neural networks, the stacking of too many graph convolution layers will lead to over-smoothing of nodes. When multiple layers are stacked, the representation of all nodes tends to be consistent, and the local structural features of nodes are lost, which is the phenomenon of over-smoothing. Therefore, most of the graph convolution network models adopt shallow structure, which will greatly limit the ability of the network to obtain high-order neighborhoods information and structural features. The common graph convolution networks usually have 2–3 layers. Take the 2-layer graph convolution network of Eq (6) as an example [44]. The first layer graph convolution aggregates the information of the first-order neighbors around the node into itself and transforms it nonlinearly, while the second-layer graph convolution aggregates the information of the second-order neighbor nodes and classifies them.

$$Z_{GCN} = softmax\left(\hat{\tilde{A}} ReLU\left(\hat{\tilde{A}} X W_0\right) W_1\right) \tag{6}$$

In recent years, a lot of research has been devoted to solving the over-smoothing problem of graph convolutional networks. Q. Li et al. [14] proved that the graph convolution of the GCN model is a special form of Laplace smoothing, where the features of a node and its neighbors are updated to new node features after a weighted average, and firstly proposed that deep GCNs are easy to cause over-smoothing. Experiments show that node embedding has been mixed on a small data set with only 5 convolution layers. In order to solve the problem of node over-smoothing, Rong et al. [45] proposed DropEdge, whose idea is to randomly delete some edges of the input graph during model training, and DropEdge can be regarded as a message reducer to reduce the over-smoothing problem caused by multi-layer graph convolution to a certain extent. Feng et al. [46] proposed stochastic neural network architecture GRAND, which reduces the sensitivity of nodes to their neighbors by randomly discarding some nodes or some features of nodes. Although the above two methods can alleviate the over-smoothing phenomenon of deep graph convolution networks, but simply deleting nodes or edges will destroy the original data features and structural integrity. In addition to the above methods, there are some works as follows that try to do deep message passing by dealing with adjacency matrices.

2.4.1 SGC. The SGC model proposed by F. Wu et al. [47] reduces the complexity of graph convolution computation by removing nonlinear transformations and folding weight matrices between continuous layers, by computing the graph convolution matrix to the K power in a single-layer neural network to capture higher-order information. Ordinary GCN convolution of each layer mainly includes three steps: node information propagation, node feature linear transformation and node nonlinear activation. For the classification task, if the Softmax function is finally used, the final classification result is Eq (7), and the specific process of graph convolution of each layer is Eq (8).

$$\hat{Y}_{GCN} = softmax\left(\tilde{P} H^{K-1} W^{K}\right) \tag{7}$$

$$H^{(t+1)} = ReLU\left(\tilde{P} H^{(t)} W^{(t)}\right) \tag{8}$$

Where $\tilde{P} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ denotes the symmetric normalized adjacency matrix, $0 \leq t \leq K$, and the input node features matrix $X = H^{(0)}$.

SGC assumes that the nonlinear operation of node features is not important to GCN, and the effect of GCN mainly comes from the feature propagation between nodes. If the nonlinear transformation of each layer is deleted, it still has the same "receptive field" as K-layer GCN. Then the above Eq (7) is converted into the following Eq (9):

$$\hat{Y}_{GCN} = softmax\left(\tilde{P} \ldots \tilde{P}\tilde{P} X W^{(1)} W^{(2)} \ldots W^{(K)}\right) \tag{9}$$

In order to simplify the above equation, the normalized adjacency matrix $\tilde{P}$ can be raised to the power of K, and the collapsed adjacency matrix $\tilde{P}^{K}$ can be obtained. At the same time, the product of multiple weight matrices can be replaced with one weight matrix, and the following simplified equation can be obtained:

$$\hat{Y}_{GCN} = softmax\left(\tilde{P}^{K} X W\right) \tag{10}$$

Through experiments, the effect of SGC is almost the same as that of GCN [2], and the speed is faster than GCN. But the premise of this method is to assume that the nonlinear transformation of node features is not important, which loses the powerful expressive ability of nonlinear structures.

2.4.2 MAGNA. G. Wang et al. [48] aimed at the limitation that one-layer graph convolution in the existing GAT [31] model can only aggregate first-order neighbors information, allowing associations between nodes that are not directly connected in the network. Based on the powers of the 1-hop attention matrix A, the attention score of multi-hop neighborhoods is calculated through the following attention diffusion process:

$$\tilde{A} = \sum_{i=0}^{\infty} \theta_i A^i \tag{11}$$

Where i represents the number of paths from node h to node t, and the maximum length is i, thus increasing the receiving field of attention, θ_i represents attention attenuation factor, and satisfies $\sum_{i=0}^{\infty} \theta_i = 1$, $\theta_i > 0$, $\theta_i > \theta_{i+1}$, i.e., the higher the order of attention factor, the smaller the weight. This method also assumes that there is a correlation between nodes that are not directly connected.

3 Deep Graph Convolutional Neural Network II (DGCNNII)

In this section, we propose DGCNNII for graph classification, which consists of four parts: 1) The graph convolution layers of the first-stage (16 layers) is used to extract the rich structure information of the input graph, and the rich high-dimensional features of the input nodes can be obtained. 2) The second-stage graph convolutional layers (16 layers) concatenates the high-dimensional node features extracted by the first-stage graph convolutional layers and the initial low-dimensional features as input to extract the deeper structural information and node features, then define a consistent vertex ordering. 3) The SortPooling layer sorts the node features output by the second-stage graph convolution layers, and unifies the number of nodes as the input of the next stage. 4) One-dimensional convolution layers and dense layers read the sorted continuous node features for graph attribute prediction. The following Fig 2 shows the architecture of DGCNNII. Table 2 summarizes the symbols used in this paper.

3.1 Non-local Message Passing Neural Network (NLMP)

Section 2 introduces two commonly used general frameworks of graph neural networks, namely MPNN and NLNN. The general framework of graph neural networks is to generalize and abstract the similar GNN networks structure, and integrate it into a unified framework, which provides ideas for flexible design and improvement of the model. The MPNN framework summarizes various GNN models and their variants from the perspective of message aggregation and updating. The NLNN framework is a generalized summary of the GNN model based on the attention mechanism. For details, see Sections 2.1 and 2.2.

Inspired by the successful application of deep neural networks in the image field, we aim to design a deep graph neural network framework which can solve the over-smoothing problem, in order to extract more remote dependencies of nodes. This framework can make the node information aggregation of graph neural network not only rely on local information, but also make vertices aggregate information from multi-hop neighborhoods by stacking multi-layer graph convolution. For graphs of different scales, the corresponding depth and information aggregation technology can be designed to aggregate the node information and structural information of the whole graph to the target node to extract higher-dimensional abstract

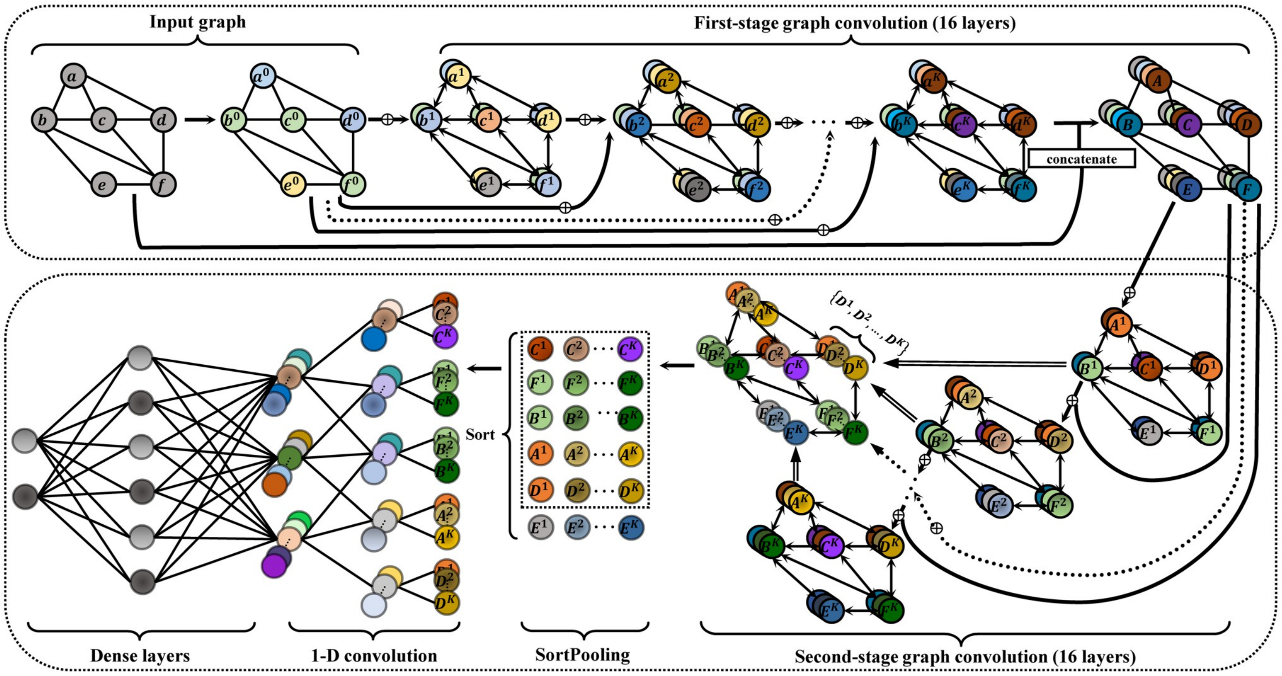
Input graph
First-stage graph convolution (16 layers)
concatenate
Sort
Dense layers
1-D convolution
SortPooling
Second-stage graph convolution (16 layers)

Fig 2. The overall structure of DGCNNII. The input graph with arbitrary structure first obtains the fine high-dimensional node features through the 16 layers of graph convolution in the first-stage, and then inputs them into the 16 layers of graph convolution in the second-stage, and the high-dimensional node features propagate deeply among the neighbors. Finally, the SortPooling layer is used to sort and intercept the nodes, and transfer them to the common convolution layer and dense layer for graph classification model training. Features are visualized as colors.

https://doi.org/10.1371/journal.pone.0279604.g002

Table 2. Table of symbols used in this paper.

Symbol	Definition
$G = (V, E)$	G: input graph, V: node set, E: edge set.
$\|V\| = n$	n: number of nodes.
$v \in V$	nodes in G.
$x_v \in R^c$	x_v: node feature vector, c: feature dimension.
$X \in R^{n \times c}$	initial feature matrix.
h_i^k	node embedding of node i in the kth layer.
$N(v)$	set of one-hop neighbors of node v in G.
$M(\cdot)$	message aggregation function.
H^k	matrix of activations in the kth layer.
W^k	a layer-specific trainable weight matrix.
$\tilde{A}$	adjacency matrix of the undirected graph G with added self-connections.
$\tilde{D}$	degree matrix of undirected graph G.
$\sigma(\cdot)$	activation function.
$f(i, j)$	a function used to calculate the attention coefficient between i and j
$g(\cdot)$	feature transformation function.
δ_k	the weight matrix decay parameter of the kth layer
Z^k	the kth layer output of DGCNNII.
$A_{GAT} \in R^{n \times n}$	adjacency matrix based on node attention coefficients.
α, β, γ	hyper-parameters for adjusting the proportion of information aggregation.
I_n	identity matrix.
a	the weight vector that projects the concatenate vector to the scalar

https://doi.org/10.1371/journal.pone.0279604.t002

features. The NLMP framework proposed in this paper is as follows:

$$h_i^{(t+1)} = \sum_{v_j \in N(v_i)} M_t\left(h_i^{(t)}, h_j^{(t)}, h_i^{(0)}, h_i^{(t-1)}, e_{ij}\right) \tag{12}$$

Compared with the Eq (1) of the MPNN framework, the above equation aggregates more information $h_i^{(0)}$ and $h_i^{(t-1)}$. For the aggregation update of target node i, it aggregates not only the information of its first-order neighboring nodes $\{h_j^{(t)} | v_j \in N(v_i)\}$ at time t, but also the initial input feature $h_i^{(0)}$ of node i and the node feature $h_i^{(t-1)}$ at the previous time. The method of aggregating $h_i^{(0)}$ and $h_i^{(t-1)}$ in the NLMP framework borrows the ideas from ResNet model and DenseNet model in the image field. The reason for introducing residual connections and dense connections in the process of node information aggregation is that when the graph neural network is deep enough, even if the node information goes through multiple rounds of iterations, some initial features $h_i^{(0)}$ of the nodes are still retained. Through back propagation learning, achieves at least the same effect as shallow networks, and the over-smoothing phenomenon is significantly reduced. The previous moment of node features $h_i^{(t-1)}$ is introduced, so that the result of each graph convolution layer always carries some node features generated by all previous layers, so that the final output node features of the model includes part of the output results of all convolutional layers. The dense connection in this paper does not connect the output results of all layers before each layer to itself, but only connects the results of the previous layer to itself, and the dependence of node features on the previous layer can be adjusted by setting parameters, it makes the adjustment of the model more flexible.

In order to investigate why graph neural networks can achieve good results, Q. Li et al. [14] compared GCNs with the simplest Fully Connected Networks (FCNs). The hierarchical propagation rules of conventional FCNs (Eq (13)) and the common neighboring information aggregation mode of GCN [2] (Eq (14)) are respectively as follows:

$$H^{(l+1)} = \sigma\left(H^{(l)} W^{(l)}\right) \tag{13}$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right) \tag{14}$$

The only difference between GCN and FCN is that GCN multiplies the normalized adjacency matrix $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ left by the feature matrix H and then right multiplies the weight matrix W. Therefore, the feature matrix processed by the normalized adjacency matrix is the reason for the good effect of graph convolution. The author also proves that graph convolution is a special form of Laplace smoothing, namely symmetric Laplace smoothing. Laplace smoothing averages the information of nodes and their neighbors as a new feature of nodes. Since the nodes in the same subgraph are often densely connected, after the import of deep graph convolution, the average aggregation of neighborhoods information makes the features of all nodes in the same subgraph tend to be consistent. And in the relational graph data, the influence of different types of neighbors on the target node is weakened. Therefore, using the idea of NLNN framework for reference, an adjacency matrix based on attention mechanism is introduced to aggregate neighbor nodes, which makes the NLMP framework proposed in this paper more generalized and scalable, easy adapts to multi-relational and single relational graph data, and makes up for the over-smoothing problem caused by average aggregation of neighbor nodes. The more specific NLMP framework design is as follows:

$$h_i^{(t+1)} = M_t\left\{\left[\frac{1}{C(h)}\sum_{v_j\in N(v_i)} f\left(h_i^{(t)}, h_j^{(t)}\right)\cdot g\left(h_j^{(t)}\right)\right], g(h_i^{(0)}), g\left(h_i^{(t-1)}\right)\right\} \tag{15}$$

In the NLMP framework, *f* represents the Gaussian function that calculates the attention coefficients between nodes, *g* represents the node feature transformation function, *M* represents the message aggregation function, and the factor $^1/_C(h)$ is used to normalize the results. The specific design of the above equation and the graph convolution layer used in this paper will be described in detail in Section 3.2.

3.2 Graph convolution layer

3.2.1 Adjacency matrix based on attention coefficient. In the image field, the similarity between pixel *i* and pixel *j* can be defined as a decreasing function of the weighted Euclidean distance, $||v(N_i) - v(N_j)||_2^2$ [29]. We transfer the concept of similarity between pixels to graph data, compare the similarity between nodes can be converted to measure the similarity of node feature vectors, and the similarity can be calculated by the inner product of node vectors after linear transformation. According to the non-local mean operation [29] and the bilateral filter proposed by Tomasi & Manduchi [49], Gaussian function can be selected:

$$f\left(h_i, h_j\right) = e^{\theta(h_i)^T\varphi\left(h_j\right)} \tag{16}$$

Where $\theta(h_i) = W_\theta h_i$, $\varphi(h_j) = W_\varphi h_j$. Based on the NLMP framework proposed in this paper, the Gaussian function used to calculate the attention coefficient between nodes adopts the concatenation function in the GAT model proposed by Velickovic et al. [31]:

$$f\left(h_i, h_j\right) = e^{LeakyRelu(a^T[Wh_i||Wh_j])} \tag{17}$$

Where h_i and h_j represent the feature vector of node *i* and node *j* respectively, $h \in R^{b\times1}$ represents the feature vector, $W \in R^{a\times b}$ represents the learnable weight matrix, and $||$ represents the concatenate operation. Therefore, $(Wh_i||Wh_j) \in R^{(2a\times1)}$. And $a^T \in R^{2a\times1}$ is the weight vector that projects the concatenate vector to the scalar. After regularizing the Gaussian function, the attention coefficient between node *i* and node *j* is obtained:

$$\frac{1}{C(h)}f\left(h_i, h_j\right) = \frac{\exp(LeakyRelu(a^T[Wh_i||Wh_j]))}{\sum_{k\in N_i}\exp(LeakyRelu(a^T[Wh_i||Wh_k]))} \tag{18}$$

3.2.2 Feature transformation based on identity mapping. For the feature transformation function *g* in Eq (15), we can use a simple linear transformation function $g(h) = Wh$, but Klicpera et al. [44] pointed out that frequent interactions between different dimensions of the feature matrix degrades the performance of the model. The GCNII model proposed by Chen et al. [50] borrows the idea of identity mapping in ResNet, and introduces the mechanism of identity mapping into GCN. The identity matrix I_n is added to the weight matrix *W*, and the weight of the identity matrix increases with the number of layers. The goal is to ensure that the deep GCN model achieves at least the same performance as the shallow GCN. Therefore, the linear transformation function in this paper is set as:

$$g(h) = \left((1-\delta_l)I_n + \delta_l W^{(l)}\right)h \tag{19}$$

In the above equation, δ_l is the weight matrix decay parameter that changes with the number of layers l. We refer to the setting $\delta_l = \log\left(^{\lambda}/_{l} + 1\right)$ in GCNII, and λ is the hyper-parameter. δ_l makes the weight matrix adaptively decay as the number of layer increases.

3.2.3 Proposed form of graph convolution. In conclusion, given an input graph $G = (V, E)$ and its node feature matrix $X \in R^{n\times c}$, the proposed DGCNNII in this paper extracts the non-local structure information of nodes by stacking deep graph convolutions to achieve the aggregation of non-local neighborhood nodes information. We define the $(t + 1)^{th}$ graph convolutional layer as:

$$Z^{(t+1)} = \sigma\left(\left(\alpha A_{GAT}^{(t)}Z^{(t)} + \beta Z^{(t-1)} + \gamma X\right)\left((1-\delta^{(t)})I_n + \delta^{(t)}W^{(t)}\right)\right) \tag{20}$$

In the above equation, $Z^{(t)}$ is the output result of the tth graph convolution layer. Initially can make $Z^{(0)} = Z^{(1)} = X$, $A_{GAT}^{(t)} \in R^{n\times n}$ is the adjacency matrix based on the attention coefficient of the t^th graph convolution layer of the input graph G, where $(A_{GAT}^{(t)})_{ij} \in \{0, r\}$. If there is an edge between node i and node j, namely $(v_i, v_j) \in E$, then $(A_{GAT}^{(t)})_{ij} = r$, the real number $r \in (0,1)$ represents the attention coefficient of the correlation between the two nodes, otherwise $(A_{GAT}^{(t)})_{ij} = 0$. details as follows:

$$\left(A_{GAT}^{(t)}\right)_{ij} = \frac{\exp(LeakyRelu((a^{(t)})^T[W_{GAT}^{(t)}h_i^{(t)}||W_{GAT}^{(t)}h_j^{(t)}]))}{\sum_{k\in N_i^{(t)}}\exp(LeakyRelu((a^{(t)})^T[W_{GAT}^{(t)}h_i^{(t)}||W_{GAT}^{(t)}h_k^{(t)}]))} \tag{21}$$

Each layer of graph convolution is divided into the following four steps: 1) Firstly, the node features after nonlinear activation are propagated through $A_{GAT}Z$ to the neighboring nodes and the nodes themselves according to different attention weights to obtain the new node feature matrix $Y = A_{GAT}Z$. 2) The new feature matrix $\bar{Y}$ is obtained by summing Y, the result of the graph convolution of the previous layer, and the initial node feature matrix according to the corresponding proportion. 3) Through $\bar{Y}((1-\delta)I_n + \delta W)$, the node feature matrix is transformed by linear feature transformation based on identity mapping. 4) Finally, the nonlinear activation function is applied to the result of the previous step, and the graph convolution result is output. Repeat the above four steps for each layer of GCN.

We suggest that the initial feature matrix X in the Eq (20) can be an one-hot vector, or it can transform the feature dimension through a fully connected neural network according to the needs of model. $Z^{(t-1)} \in R^{n\times c_{t-1}}$ is the output of the $(t - 1)^{th}$ graph convolutional layer, c_{t-1} is the number of output feature channels of the node in the $(t - 1)^{th}$ layer, The weight matrix $W^{(t)} \in R^{c_t\times c_{t+1}}$ is used to map the c_t node feature channels into c_{t+1} feature channels. β and γ are parameters that control the proportion of the previous layer output and initial node features, which can be adjusted manually. α can be simply set to $\alpha = 1 - \beta - \gamma$. In this paper, both β and γ are set to 0.1, and σ is the nonlinear activation function. After the second-stage graph convolution of the model is completed, a layer is added to connect the output result $Z^{(t)}$ ($t = 1, \ldots, K$) of each graph convolution layer, denoted as $Z^{1:K} = [Z^{(1)}, Z^{(2)}, \ldots, Z^{(K)}]$, each row of the concatenated output $Z^{1:K}$ can be viewed as a "multi-scale feature vector" of a vertex and used as input to the remaining layers.

Table 3. Common graph pooling methods.

Graph pooling methods	Graph pooling equation
SumPooling	$r^{(i)} = \sum_{k=1}^{N_i} x_k^{(i)}$
AvgPooling	$r^{(i)} = \frac{1}{N_i}\sum_{k=1}^{N_i} x_k^{(i)}$
MaxPooling [51]	$r^{(i)} = max_{k=1}^{N_i}(x_k^{(i)})$
GlobalAttentionPooling [52]	$r^{(i)} = \sum_{k=1}^{N_i} softmax(f_{gate}(x^{(i)}k))f_{feat}(x_k^{(i)})$

https://doi.org/10.1371/journal.pone.0279604.t003

3.3 Remaining layers

3.3.1 Graph pooling layer. The graph pooling layer is used to aggregate the node features learned by the graph convolution layer for subsequent operations. Common graph pooling methods in graph neural network models include: TopKPooling, SAGPooling, Set2Set, etc. Table 3 lists some other commonly used graph pooling methods and specific operations. $x_k^{(i)}$ represents the ith dimension of node embedding, N_i represents the neighbor nodes of node i.

In order to compare with the DGCNN framework of Zhang et al. [21], and to reflect the advantages of Deep Graph Convolution Neural Network II (DGCNNII) proposed in this paper, the layers behind the graph convolution layers adopt the same configuration as DGCNN. For details, please refer to the papers of Zhang et al. The graph pooling of this paper uses SortPooling. As mentioned by Zhang et al., the output of the graph convolution layer in this paper is also a continuous WL (Weisfeiler-Lehman) color, and the deeper the convolution layer is, the more $Z^{(K)}$ can divide the node into different colors / groups. The DGCNN model has only 4 layers of graph convolution, and the DGCNNII model proposed in this paper has a depth of 32 layers. The nodes are sorted in descending order according to the last channel $Z^{(K)}$ output by the graph convolution layer. If the values of two nodes are the same in the $Z^{(K)}$ channel, they are compared through the previous layer $Z^{(K-1)}$, and so on. The output of the graph convolution layers is a tensor of shape $n \times \sum_1^K c_t$ as the input of the SortPooling layer, because the number of nodes in each subgraph is different, in order to facilitate the subsequent processing of the network, the SortPooning layer truncates / expands the input tensor of $n \times \sum_1^K c_t$ to the tensor of $k \times \sum_1^K c_t$ size.

3.3.2 Traditional layers. Consistent with the paper of Zhang et al., first convert the $k \times \sum_1^K c_t$ tensor output by the SortPooling layer into a row vector of $k(\sum_1^K c_t) \times 1$, and then add maximum pooling layers, one-dimensional convolution layers, fully connected layers, and a softmax layer, etc.

4 Discussion

As mentioned earlier, we need the graph convolution layers deep enough to divide the nodes into different groups / categories as much as possible. However, when the graph convolutional layers are stacked too much, there will be serious over-smoothing problem. In a densely connected graph, each node has many common neighbor nodes, if there are too many layers in the graph convolutional neural network, the number of aggregated neighbor nodes will increase and the number of overlapping nodes will increase, which will easily lead to the consistency of the final node feature representation, and the features of different nodes will be covered up.

The DGCNNII model proposed on the basis of the NLMP framework provides a solution to the above problems and achieves state-of-the-art results. This framework mainly solves the problem that the node features tend to be consistent caused by the average aggregation of a large number of overlapping neighbor nodes. The graph convolution in DGCNNII does not simply use the normalized adjacency matrix to aggregate neighbor information according to the degree of nodes, but introduces the graph attention mechanism to aggregate information according to the different attention coefficients between nodes and their neighbors to adaptively learn the weights of nodes with different importance to avoid average aggregation leading to the consistency of node features. At the same time, residual connections and dense connections are introduced to make each information aggregation includes the initial features of nodes and the results of the previous graph convolution layer, ensuring that the model achieves at least the results of the shallow version. In order to better eliminate the over-smoothing phenomenon, the identity mapping mechanism is also introduced. DGCNNII's double-stage graph convolution framework can extract rich node features for deep node information propagation, combined with the above techniques can achieve amazing results.

5 Experiments

In this section, we use 5 bioinformatics datasets to validate the performance of DGCNNII on graph classification task. And compared with the classical graph kernel methods and other deep learning methods. In order to reflect the effect of DGCNNII on eliminating the over-smoothing phenomenon, we quantify the graph smoothness of each graph convolutional layer of the model and compare it with DGCNN. Ablation studies were also carried out to demonstrate the effectiveness of the model architecture. Finally, we analyze the model based on the experimental results. All experiments run on a computer with 12-core Intel(R) i7-12700KF CPU, 16 GB RAM, and NVIDIA Geforce RTX 3080 12GB GPU. We use Pytorch to implement our methods.

5.1 Experimental setup

5.1.1 Dataset. Because DGCNNII has the advantage of extracting rich node information based on non-local structural features, this paper uses 5 bioinformatic datasets with node labels, named MUTAG [53], PTC [54], PROTEINS [55], D&D [56], NCI1 [57], instead of using purely structural datasets without node labels, the initial node features of each bioinformatics dataset are represented by one-hot vectors.

1. The MUTAG dataset consists of 188 compounds, which are divided into two categories according to their mutagenic effect on bacteria. Each compound represents a graph, and each atom represents the nodes in the subgraph.
2. The PTC dataset also consists of 344 compounds (graphs), which are divided into two categories according to whether they are carcinogenic to mice. Compounds are composed of 19 kinds of atoms (nodes).
3. The PROTEINS dataset consists of 1113 protein structure graphs, all protein structures (graphs) are divided into two classes of enzymes/non-enzymes, and nodes consist of 3 classes.
4. The D&D dataset consists of 1178 protein structure graphs, all protein structures (graphs) are classified into enzymatic/non-enzymatic categories, and the nodes are composed of 82 amino acids.
5. NCI1 is a cancer cell activity screening compound dataset, and the graph labels are divided into two categories with/without anticancer activity.

The following table summarizes the specific parameters of the above datasets.

5.1.2 Model settings. In order to mainly compare the DGCNN model and highlight the advantages of the deep graph convolution proposed in this paper, except for the graph convolution layers, the experimental settings of other parts of the model refer to DGCNN. In order to achieve a fair comparison, we follow to set up the experiments for 10 times and report the average test accuracy using the 10-fold cross validation. The graph convolution of DGCNNII has two stages, and each stage has 16 graph convolution layers. The first 15 layers of graph convolution in the first-stage all have 128 output channels, and the number of output channels of the 16th layer of graph convolution is set differently according to different data sets. In this paper, the number of output channels (output feature dimension) of the 16th layer is set to 32, 64, 128, 32, 128 for the MUTAG, PTC, NCI1, PROTEINS, D&D datasets, respectively. The output feature matrix of the 16th layer of DGCNNII for each of the above datasets also contains the initial one-hot vector of the node. The operation of concatenating the initial feature vector of nodes after the output of the first-stage graph convolution can effectively reduce the phenomenon of node over-smoothing. The quantitative measurement of the graph smoothness of each layer of the model in Section 5.3 can show the effectiveness of concatenating operation. In particular, there is a fully connected layer that converts the initial input one-hot vector into 128 dimensions before the first-stage of graph convolution, which is used for feature dimension conversion. The second-stage graph convolution consists of a fully connected layer which converts the input feature dimension into 128 dimensions and 16 graph convolution layers. The first 15 layers of graph convolution have 128 output channels, and the last layer convolution is a single channel output. We use the one-channel features output from the last graph convolutional layer for node sorting. The remaining layers are composed of two one-dimensional convolution layers and one dense layer. The configurations of the two one-dimensional convolution layers are: 1) Filter size is 2, maximum pooling layer with step size 2, output channel is 16. 2) Filter size is 5, step size is 1, output channel is 32. The dense layer has 128 hidden units, and finally the Softmax layer.

5.1.3 Parameter settings. The α, β and γ in Eq (20) are set to 0.8, 0.1 and 0.1 respectively. The λ in the weight matrix attenuation parameter $\log\left(\frac{\lambda}{l}+1\right)$ is set to 0.5. A dropout layer with dropout rate 0.5 is used after the dense layer. The training batch size is set to 1, and the partition ratio of the training sets and the test sets is 9:1. Rectified linear units (ReLU) is used as a nonlinear function in the convolution layer and other layers. Stochastic gradient descent (SGD) with the ADAM updating rule [58] was used for optimization. In order to fairly compare the performance of DGCNN and DGCNNII in the graph classification task, and highlight the improvement brought by the use of deep graph convolutional layers, we use the same learning rate and epoch number as DGCNN in the MUTAG, PTC, NCI1, PROTEINS, D&D dataset without hyper-parameter adjustment.

5.2 Comparison with other graph classification methods

5.2.1 Comparison with graph kernel. We compare DGCNNII with the following 6 classical graph kernel methods: Shortest-Path Kernel [59], the Graphlet Kernel [60], the Random Walk Kernel [61], the Weisfeiler-Lehman subtree kernel [62], the Propagation Kernel [63] and Deep Graph Kernels [64]. For fair comparison, we use a single network structure on all datasets. We compare the results of DGCNNII with methods based on graph kernel mentioned above and take the reported accuracy directly from their papers ("–" means not available). The results are shown in Table 4, and the best results are shown in bold.

From the results in Table 4, we can see that although DGCNNII uses the same single structure on all data sets, but still achieves excellent results compared with the kernel results. The accuracy on MUTAG, PTC, PROTEINS, and D&D are higher than that of all graph kernel

Table 4. Comparison of graph classification accuracy with graph kernel methods.

Methods	Dataset				
	MUTAG	PTC	NCI1	PROTEINS	D&D
SP	87.28±0.55	58.24±2.44	73.47±0.11	75.07±0.54	78.86± 0.26
GK	81.39±1.74	55.65±0.46	62.49±0.27	71.39±0.31	74.38±0.69
RW	79.17±2.07	55.91±0.32	>3 days	59.57±0.09	>3 days
WL	84.11±1.91	57.97±2.49	**84.46±0.45**	74.68±0.49	78.34±0.62
PK	76.00±2.69	59.50±2.44	82.54±0.47	73.68±0.68	78.25±0.51
DGK	87.44±2.72	60.08±2.55	80.31±0.46	75.68±0.54	-
DGCNNII (proposed)	**94.44**	**76.47±2.94**	80.32±0.76	**82.88±0.83**	**83.33±1.29**

https://doi.org/10.1371/journal.pone.0279604.t004

baseline methods, and DGCNNII has achieved a significant improvement on MUTAG and PTC datasets with smaller graph sizes. It shows that the two small-scale datasets after 32-layer depth graph convolution, the nodes aggregate more comprehensive non-local information, which greatly improves the prediction accuracy.

5.2.2 Comparison with other graph neural network models. We compare DGCNNII with a variety of graph classification baseline methods based on deep learning in recent years, including various models based on MPNN and NLNN framework. Among them, Diffusion-CNN (DCNN) [65] and PATCHY-SAN [66] both extend convolutional neural networks (CNNs) to general graph structure data for graph convolution operation. The convolution operation of ECC [67] and the conventional two-dimensional image convolution are both weighted average operation, but ECC can be applied to any graph structure, and the weight is determined by the edge weight between nodes. One-head graph attention network (1-head GAT) [31] can assign different weights to different nodes in the neighborhoods, rather than simply weighted average neighbor nodes. GCAPS-CNN is a graph capsule network proposed by Verma & Zhang [68]. AWE [69] proposes two anonymous sequence graph representation methods based on feature vector and embedding vector. S2S-N2N-PP [70] generates the node sequence of the graph through methods such as random walk, breadth first search, and shortest path, and then feeds it into a long short-term memory (LSTM) autoencoder for training to obtain a vector representation. The Motif-based filtering in the NEST [71] model can capture the fine structures in the network, while the convolution based on filtering embedding enables it to fully explore complex substructures and their combinations, thus carrying out graph classification tasks. CapsGNN [72] proposes a new vector propagation mode and hierarchical prediction, which makes the network more interpretable. GIN [17] proposes a simple architecture and has the same powerful graph isomorphism recognition capability as the Weisfeiler-Lehman graph kernel. MA-GCNN [73] is a new Motif-based attention graph convolutional neural network that can learn more discriminative and richer graph features. This paper also compares the baseline methods of four improved graph pooling functions, such as DiffPool [27], gPool [43], EigenPool [41] and SAGPool [42]. The specific comparison results are shown in Table 5. The best results are shown in bold.

We can see that DGCNNII using a single structure surpasses all the baseline methods in terms of accuracy with only one exception on NCI1. It is worth mentioning that since MUTAG has only 188 graphs, under the 9:1 training / test set division, the test set has only 18 graphs. When the categories of 17 graphs are correctly predicted, the accuracy rate reaches 94.44%. However, in the actual prediction process of DGCNNII, a large number of cross-validation reached 100% accuracy, and it can be seen from Fig 5 that when the training reached the 20th epoch, it had reached 100% accuracy, and continued to stabilize at more than 94%, while maintaining lower loss and higher AUC than DGCNN. DGCNNII also achieves the best

Table 5. Comparison of graph classification accuracy with other graph neural network based methods.

Methods	Dataset				
	MUTAG	PTC	NCI1	PROTEINS	D&D
DCNN			56.61±1.04	61.29±1.60	58.09±0.53
PATCHY-SAN	92.63±4.21	62.29±5.68	76.34±1.68	75.00±2.51	76.27±2.64
ECC	89.44	-	76.82	-	72.54
GAT(1-head)	81.0	57.0	74.3	72.5	-
GCAPS-CNN	-	66.01±5.91	82.72±2.38	76.40±4.17	77.62±4.99
AWE	87.87±9.76	-	-	-	71.51±4.02
S2S-N2N-PP	89.86±1.10	64.54±1.10	**83.72±0.40**	76.61±0.50	-
NEST	91.85±1.57	67.42±1.83	81.59±0.46	76.54±0.26	78.11±0.36
CapsGNN	86.67±6.88	-	78.35±1.55	76.28±3.63	75.38±4.17
GIN	89.40±5.60	64.60±7.00	82.70±1.70	76.20±2.80	-
MA-GCNN	93.89±5.24	71.76±6.33	81.77±2.36	79.35±1.74	81.48±1.03
DiffPool	-	-	-	76.25	80.64
gPool	-	-	-	77.68	82.43
EigenPool	-	-	77.00	76.60	78.60
SAGPool	-	-	67.45±1.11	71.86±0.97	76.45±0.97
DGCNN	85.83±1.66	58.59±2.47	74.44±0.47	75.54±0.94	79.37±0.94
DGCNNII (proposed)	**94.44**	**76.47±2.94**	80.32±0.76	**82.88±0.83**	**83.33±1.29**

https://doi.org/10.1371/journal.pone.0279604.t005

results on PTC, with an average accuracy rate of 4.71%-19.47% higher than other baseline methods. Although not achieving the best accuracy on NCI1, but the accuracy rate is about 6% higher than DGCNN. While all other baseline methods have not reached 80% accuracy on PROTEINS dataset, DGCNNII achieves 81.08% accuracy. The highest accuracy is also achieved on the D&D dataset.

We analyze the NCI1 dataset. According to Table 6, we can see that although the NCI1 dataset has the largest number of subgraphs, but it is the only dataset with a higher number of node labels than the average number of subgraph nodes in the 5 datasets, which means that many subgraphs in NCI1 do not fully contain all types of nodes. While our proposed deep graph convolutional model uses the strategy of node information aggregation and update, although the deep network model can aggregate higher-order node information, due to the lack of information in the subgraphs of the dataset, all node features are not covered during training, so it cannot achieve the best results on the test set. We also compare and analyze the S2S-N2N-PP model which has the best classification result on NCI1. The method is to generate sequences from graphs by random walk, breadth-first search and shortest path, and then use recurrent neural network automatic encoder to embed graph sequences into continuous vector space to learn graph representation. S2S-N2N-PP does not adopt the information

Table 6. Common datasets for graph classification tasks.

Dataset	Graphs	Classes	Nodes(max)	Nodes(avg.)	Node Labels
MUTAG	188	2	28	17.93	7
PTC	344	2	109	25.56	19
NCI1	4110	2	111	29.87	37
PROTEINS	1113	2	620	39.06	3
D&D	1178	2	5748	284.32	82

https://doi.org/10.1371/journal.pone.0279604.t006

aggregation and update strategy of traditional GCN, and does not depend on the comprehensiveness of the training set, so it has achieved good results on the NCI1 data set.

5.3 Quantitative analysis the smoothness of graph nodes representations

In order to solve the problem of node over-smoothing caused by stacking too many layers of GCN, we redefine the widely used message passing neural network (MPNN) framework and propose a new non-local message passing framework (NLMP). Based on NLMP, a 32-layer deep graph convolutional neural network model DGCNNII is designed for graph classification tasks. This section will use a quantitative method to measure the graph smoothness of each layer of the model, and the quantitative method used is the quantitative metric Mean Average Distance (MAD) proposed by Chen et al. [74]. We deepen the DGCNN model from 4 layers to 32 layers, and compare the graph smoothness between the DGCNN and the 32-layer DGCNNII proposed in this paper. Through the experimental results of Figs 3 and 4, we can see that the MAD value of each layer of the DGCNNII model is significantly higher than that of the DGCNN model (the higher the MAD value is, the lower the smoothness of the graph representation is). It is quantitatively proved that the method proposed in this paper can effectively eliminate the over-smoothing problem caused by the deepening of GCN.

5.3.1 MAD: Metric for smoothness. MAD reflects the smoothness of graph representation by computing the average of the average distances from nodes to other nodes in the graph. Since graph smoothness refers to the similarity of graph node representation, so the node similarity is measured by calculating the average distance between nodes. The equation

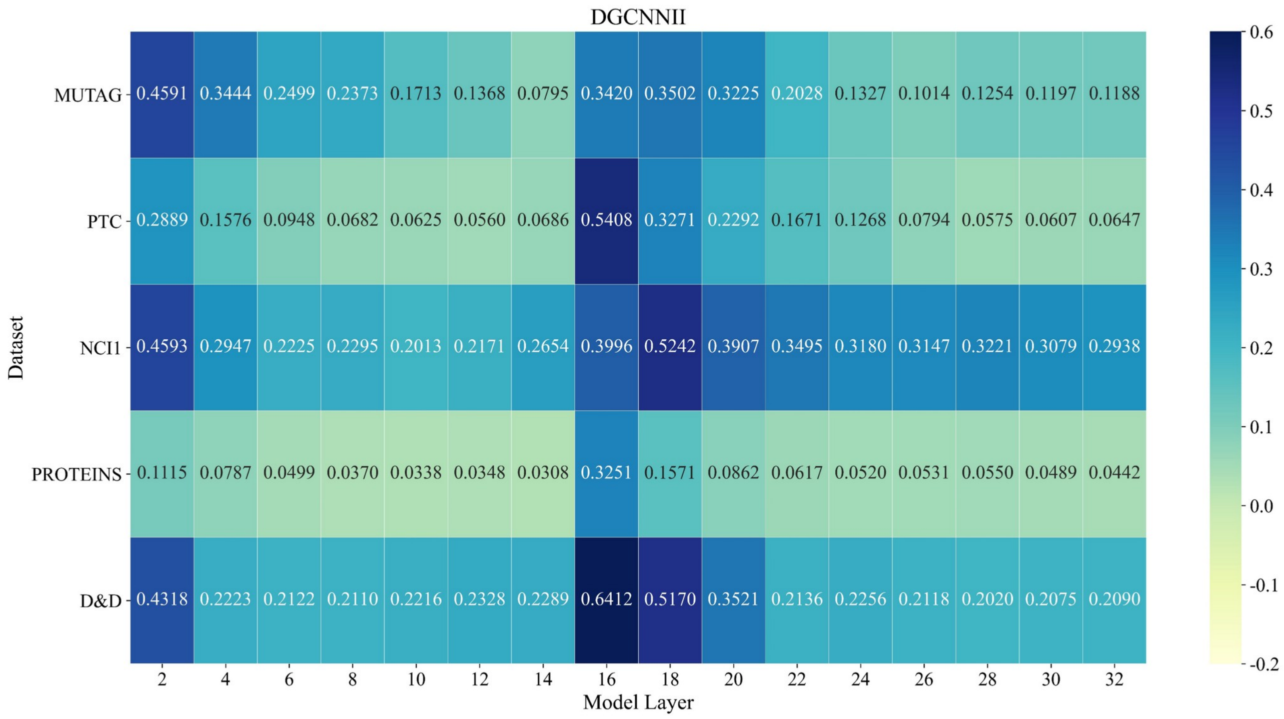

Fig 3. MAD values of different layers of DGCNNII on 5 datasets. Darker color means larger MAD value. We can find that at the output of the first-stage of DGCNNII (layer 16), the MAD value is greatly improved.

https://doi.org/10.1371/journal.pone.0279604.g003

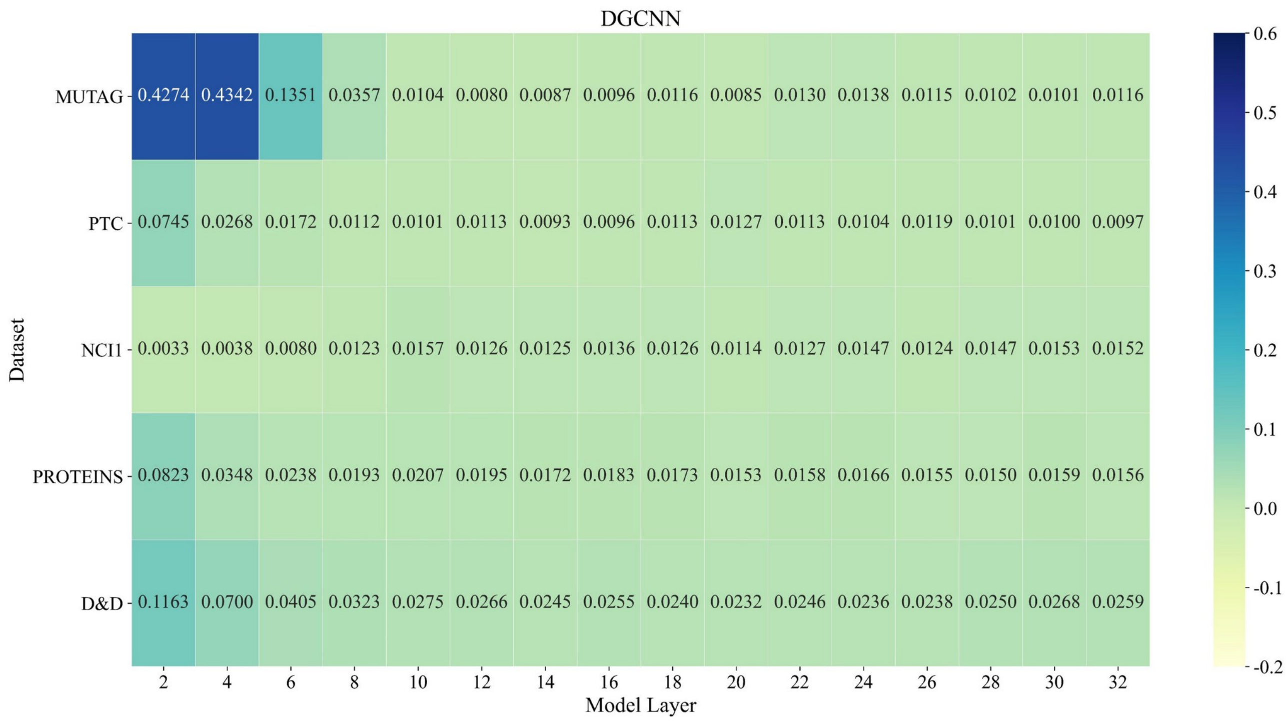

Fig 4. MAD values of different layers of 32-layer DGCNN on 5 datasets. Lighter color means smaller MAD value. We can find that the smoothness of the graph representation increases with the number of model layers, and the over-smoothing phenomenon of DGCNN is significantly more serious than that of DGCNNII.

https://doi.org/10.1371/journal.pone.0279604.g004

for calculating the MAD value of the target node pair is:

$$MAD^{tgt} = \frac{\sum_{i=0}^{n} \bar{D}_i^{tgt}}{\sum_{i=0}^{n} 1\left(\bar{D}_i^{tgt}\right)} \tag{22}$$

Where 1(x) = 1 if x > 0 otherwise 0, that is, the MAD values of the target node pairs are the average of the non-zero values in $\bar{D}_i^{tgt}$. The equation of $\bar{D}_i^{tgt}$ and its parameters is:

$$\bar{D}_i^{tgt} = \frac{\sum_{j=0}^{n} D_{ij}^{tgt}}{\sum_{j=0}^{n} 1\left(D_{ij}^{tgt}\right)} \tag{23}$$

$$D^{tgt} = D \circ M^{tgt} \tag{24}$$

$$D_{ij} = 1 - \frac{H_{i,:} \cdot H_{j,:}}{|H_{i,:}| \cdot |H_{j,:}|} \quad i, j \in [1, 2, \ldots, n] \tag{25}$$

The above Eq (23) means to average the non-zero elements in each row of D^{tgt}. $M^{tgt} \in R^{n\times n}$ in Eq (24) represents the mask matrix with the same shape as the adjacency matrix, and ∘ represents filtering the distance matrix $D \in R^{n\times n}$ by element-wise multiplication. Eq (25) represents the elements in D, D is obtained by calculating the cosine value between each node pairs, and $H \in R^{n\times h}$ is the graph representation matrix, where n is the number of nodes in the graph, term h is the hidden size. $H_{k,:}$ represents the kth row of H. We take the output of the last layer of the model as H.

It should be noted here that in the paper of Chen [74] et al., all node pairs in the graph are considered to calculate the MAD value, that is, $M^{tgt} \in R^{n \times n}$ is an all-one matrix. We consider that the diagonal matrix represents the node pair between the node and itself, and due to the characteristics of some data sets, the adjacent nodes are similar. So in the process of calculating the MAD value, the diagonal of M^{tgt} and the position of the adjacent nodes are all set 0, the rest of the positions are set to 1 to calculate the local smoothness of the graph representation. We use the same computational criteria in all experiments without affecting the objective comparison of experiments. Fig 3 shows the MAD values of different layers of our proposed 32-layer DGCNNII model on 5 datasets, and Fig 4 shows the MAD values of the deepened 32-layer DGCNN model on 5 datasets on different layers. The darker color in the figures means a larger MAD value, which means less similarity between the nodes in the graph, otherwise the more similar.

By quantitatively comparing the smoothness of each layer of the 32-layer DGCNNII and DGCNN model, we find that the smoothness of the graph representation of DGCNNII at the 32nd layer is even lower than that of the 2–4 layer of DGCNN, which indicates that the deep graph convolution network construction method proposed by us can effectively reduce the phenomenon of over-smoothing. The subsequent ablation studies in this paper will also prove that the smoothness of graph representation is not the decisive factor affecting the result of graph classification task, the structure and depth of graph convolution network model can also affect the result of graph classification.

5.4 Ablation study

We will conduct ablation studies on the necessity of the double-stage structure of the DGCNNII and why the model chooses 32 layers.

5.4.1 The necessity of double-stage model structure. In order to verify the function of the 16 layers of the second stage, we delete the last 16 layers of the DGCNNII model in Fig 2, leaving only the first 16 layers of the first stage. The graph classification results of the deleted model on the 5 data sets are shown in the first row "Only first stage (16layers)" of Table 7. In order to verify the function of 16 layers in the first stage, we change the double-stage model structure of DGCNNII into a 32-layer single stage model structure, and the graph classification results on 5 data sets are shown in the second row "Only one stage (32layers)" of Table 7. By comparing the experimental results of the above two modified models with the double-stage structure DGCNNII (32layers) model proposed in this paper, we can see that the result of the double-stage structure model is better than that of the single-layer structure model, so it is necessary to apply the double-stage structure model of DGCNNII.

Table 7. Experimental results of ablation study of DGCNNII model.

Methods (DGCNNII)	Dataset				
	MUTAG	PTC	NCI1	PROTEINS	D&D
Only first stage(16 layers)	92.11±0.87	74.76±2.51	78.35±0.76	81.08±0.54	82.05±0.80
Only one stage(32 layers)	92.11±0.91	70.82±2.57	75.91±0.76	80.18±0.81	82.05±0.83
DGCNNII-18(16+2 layers)	84.21±0.85	70.71±2.64	76.64±0.75	80.18±0.79	81.20±0.82
DGCNNII-22(16+6 layers)	92.11±0.91	68.88±2.54	78.59±0.75	79.28±0.79	81.20±0.82
DGCNNII-26(16+10 layers)	89.47±0.89	72.71±1.63	77.37±0.74	78.38±0.78	82.05±0.83
DGCNNII-30(16+14 layers)	94.44	74.59±2.66	79.32±0.79	81.98±0.82	82.91±0.84
DGCNNII(32 layers)	**94.44**	**76.47±2.94**	**80.32±0.76**	**82.88±0.83**	**83.33±1.29**

https://doi.org/10.1371/journal.pone.0279604.t007

5.4.2 Comparison of models with different layers. In order to compare the graph classification results of DGCNNII model with different layers, we carried out ablation studies for different layers of the model. Through Fig 3, we can see that the node feature of the first stage output of DGCNNII (layer 16) has a very low smoothness (higher MAD value), and through Table 7, we can also see that the DGCNNII with only first stage has a good graph classification accuracy. Therefore, we decided to adjust the number of layers in the second-stage to carry out the ablation study on the basis of retaining the first-stage. The specific experimental results are shown in Table 7 below. For example, "DGCNNII18 (16+2layers)" means adding 2 layers of second-stage graph convolution to the 16-layer of first stage, and "DGCNNII (32layers)" represents the 32-layer deep graph convolutional model (Fig 2) finally adopted in this paper. The experimental results show that the 32-layer DGCNNII achieves better results than the shallow version.

5.5 Comparison and analysis with DGCNN

In this paper, we propose a Non-local Message Passing (NLMP) framework, and a novel deep graph convolution layer and DGCNNII model based on NLMP are proposed. The DGCNNII model replaces the graph convolution part with our proposed deep graph convolution on the basis of the DGCNN model. The latter graph pooling layers and traditional layers are consistent with DGCNN. The purpose is to validate the advantages of the NLMP framework by comparing the performance of the two models on the graph classification task. Next, we analyze the performance of DGCNNII and DGCNN on the same dataset.

5.5.1 Classification accuracy comparison. Classification accuracy (Accuracy) is used to measure the proportion of correctly classified graphs in all graphs. The following Figs 5–9

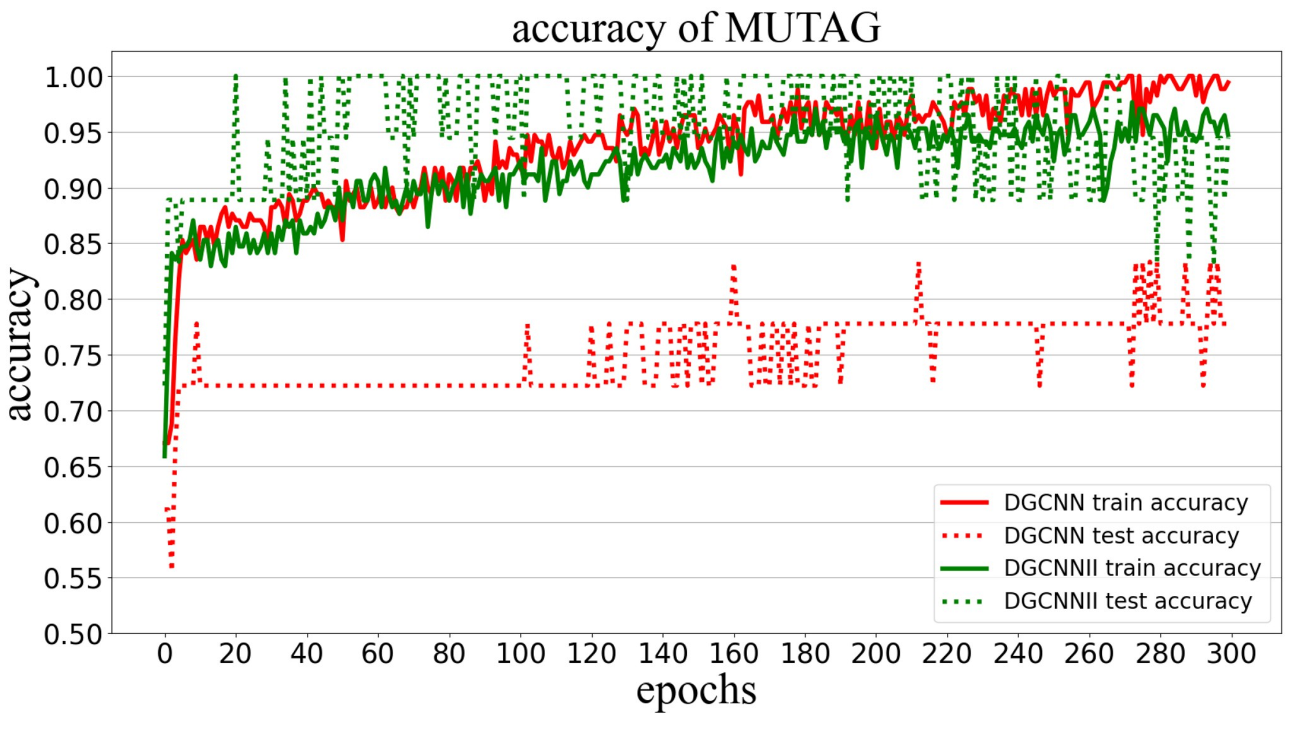

Fig 5. Comparison of graph classification accuracy between DGCNNII and DGCNN on MUTAG dataset.

https://doi.org/10.1371/journal.pone.0279604.g005

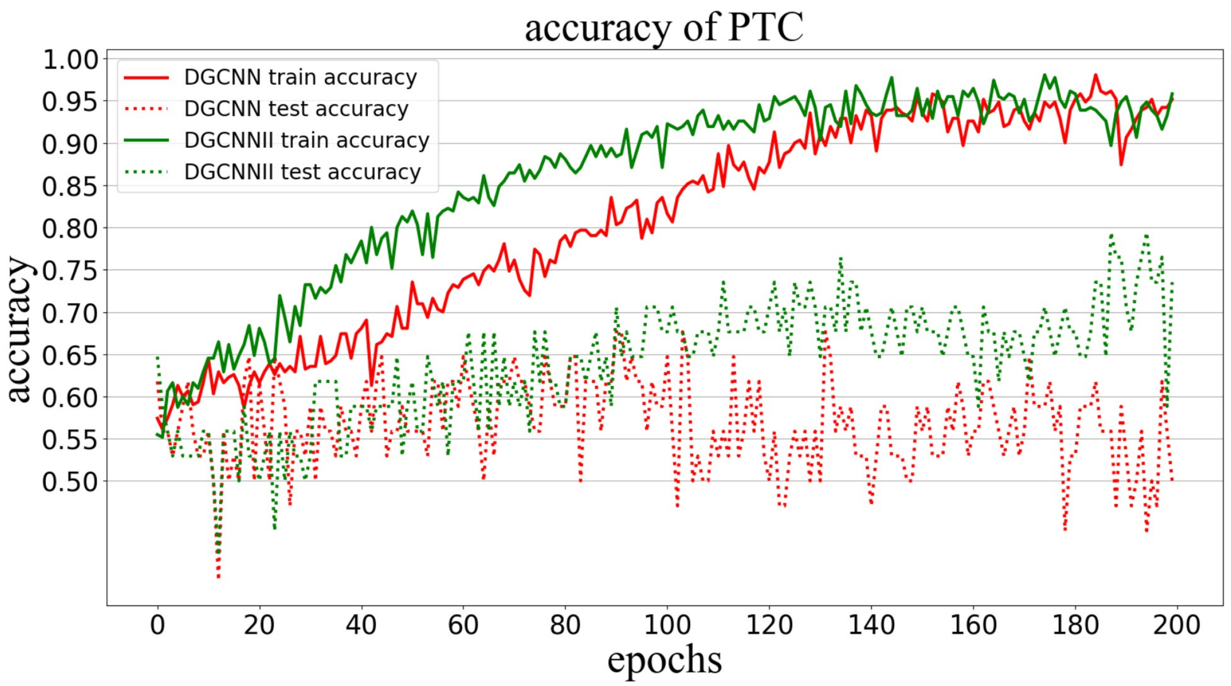

Fig 6. Comparison of graph classification accuracy between DGCNNII and DGCNN on PTC dataset.

https://doi.org/10.1371/journal.pone.0279604.g006

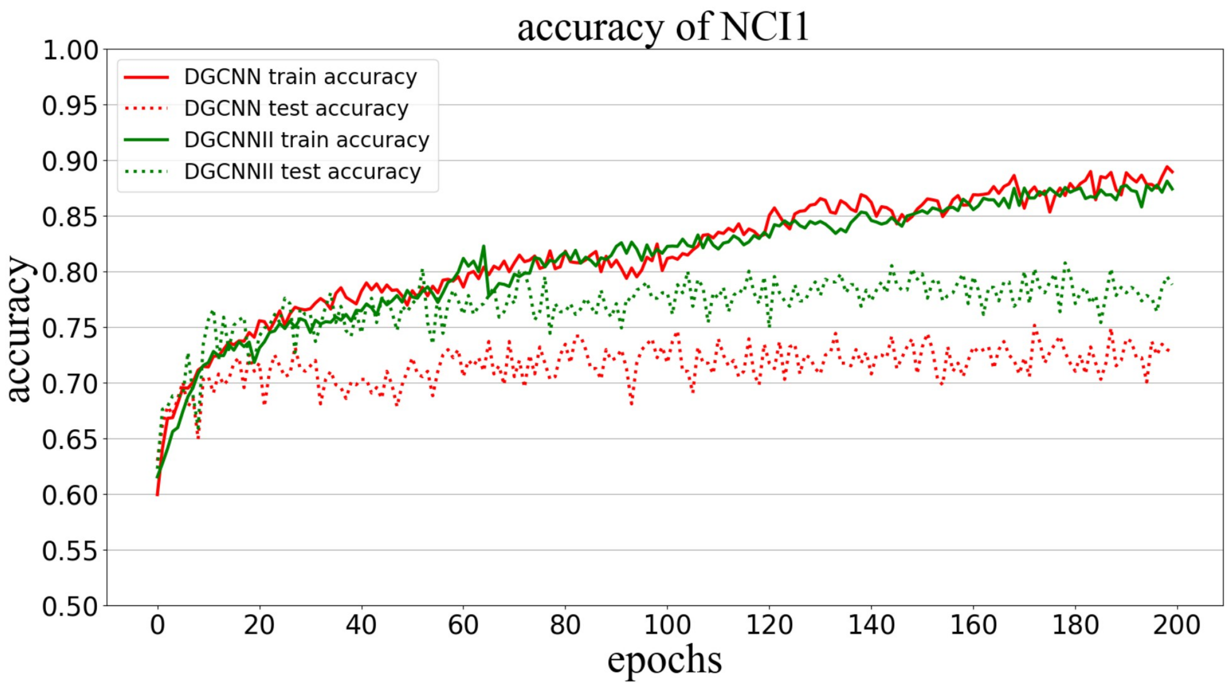

Fig 7. Comparison of graph classification accuracy between DGCNNII and DGCNN on NCI1 dataset.

https://doi.org/10.1371/journal.pone.0279604.g007

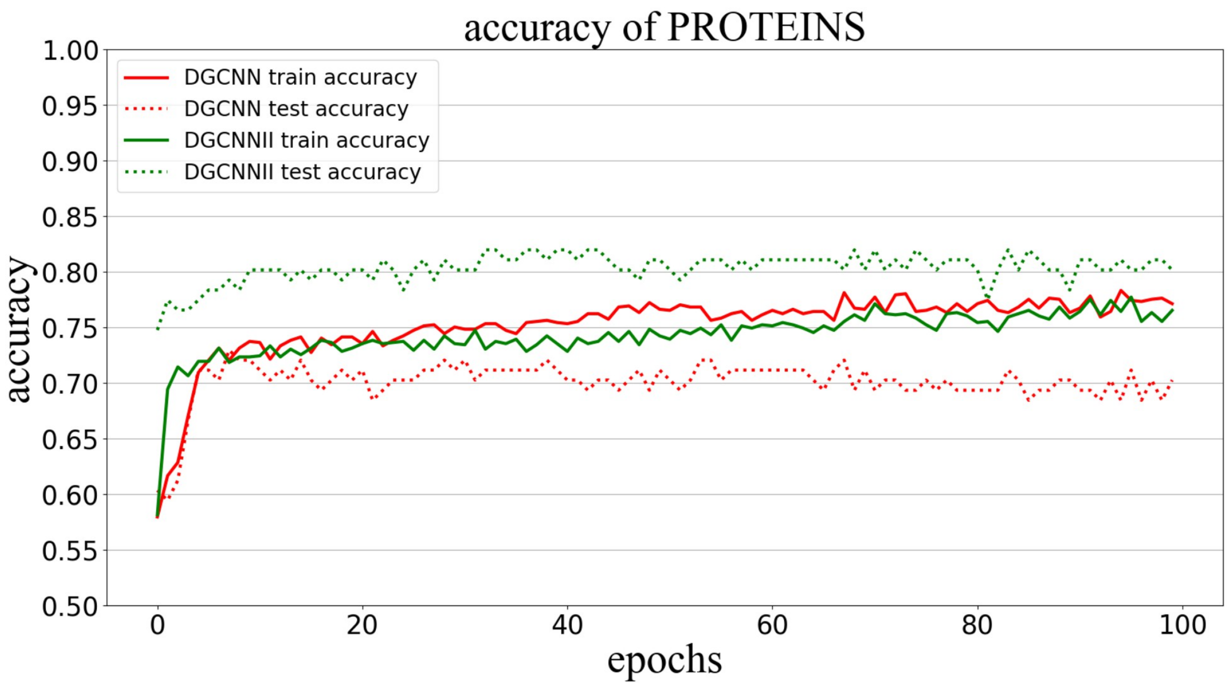

Fig 8. Comparison of graph classification accuracy between DGCNNII and DGCNN on PROTEINS dataset.

https://doi.org/10.1371/journal.pone.0279604.g008

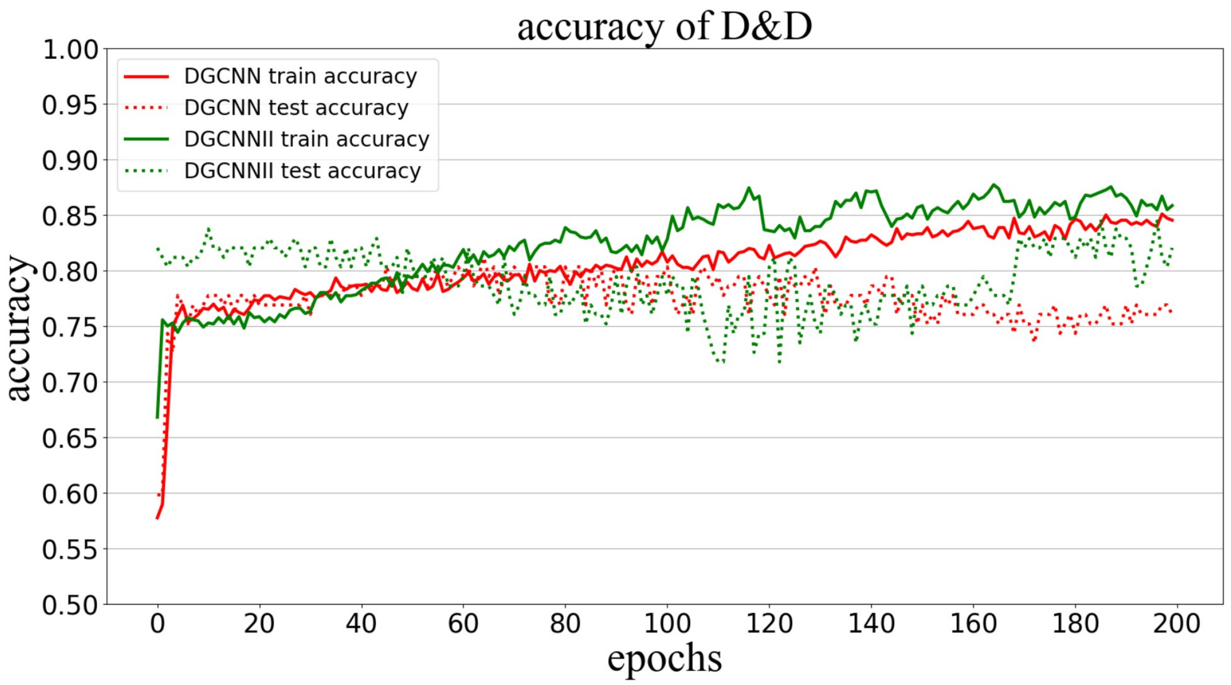

Fig 9. Comparison of graph classification accuracy between DGCNNII and DGCNN on D&D dataset.

https://doi.org/10.1371/journal.pone.0279604.g009

show the classification accuracy curves of DGCNNII and DGCNN on 5 datasets with the change of training epochs.

As can be seen from the Fig 5, on the MUTAG data set, after the 20th round of DGCNNII training, the accuracy of the test set can reach 100%, and then remain between 94% and 100%. However, DGCNN can only achieve the highest accuracy of about 85% in the 160th round of training, and the accuracy of the test set of DGCNN is far lower than the accuracy of the training set, indicating that the DGCNN model has the phenomenon of over-fitting. However, from the accuracy curve of DGCNNII training set and the accuracy curve of test set in Fig 5, we can see that the difference of accuracy between test set and training set in the later epochs are very small, indicating that DGCNNII greatly **reduces the over-fitting phenomenon**.

As can be seen from the Fig 6, on the PTC data set, although the accuracy of the training set of the two models has been increasing, but the accuracy of the test set of DGCNN decreased after 100 epochs of training, while the accuracy of DGCNNII has been steadily increasing, indicating that DGCNNII is **more stable**.

As can be seen from the Fig 7, on the NCI1 data set, the accuracy of the training / test set of the two models has been steadily increasing. Although the accuracy of the training set of the two models is almost the same, but the accuracy of the test set of DGCNNII is always higher than that of DGCNN.

On the PROTEINS data set, it can be seen that the accuracy of the test set of DGCNNII is continuously stable and much higher than that of DGCNN.

On the D&D data set, the accuracy of the test set of DGCNN continues to decrease after the 140th epoch of training, while the accuracy of the training set continues to increase, indicating that over-fitting occurred, but the accuracy of the test set of DGCNNII continues to increase, its stability and the ability to reduce over-fitting are further verified.

5.5.2 Comparison of generalization ability. The generalization ability of the model in deep learning refers to the adaptability of the model to unknown samples, which can be measured by the classification accuracy of the test set. However, it should be noted that comparing the generalization ability of the two models needs to meet the following three conditions: 1) Both models are trained in the same training set. 2) The test set is unknown sample data. 3) The test set and the training set belong to the same distribution of data. All the experiments in this paper are carried out under the condition of meeting the above three conditions, so we can directly compare the generalization ability of the two models by comparing the graph classification accuracy of DGCNNII and DGCNN on 5 test data sets. As can be seen from Figs 5–9, the accuracy of DGCNNII on the 5 test data sets is higher than that of DGCNN, so DGCNNII has **stronger generalization ability** than DGCNN.

In the field of GNNs, with the stacking of a large number of graph convolution layers, there will be serious over-smoothing problem. This section makes an experimental comparison between the DGCNN model with 4 graph convolution layers and the DGCNNII model with 32 graph convolution layers. It can be clearly seen that the DGCNNII model based on the NLMP framework not only solves the problem of over-smoothing, but also exerts the ability of deep learning to extract abstract features, reduces the phenomenon of over-fitting, and maintains the accuracy higher than the general baseline method. It also has strong stability and generalization ability.

5.5.3 Compare Area Under ROC Curve (AUC) and loss. AUC can be used as an indicator to measure the quality of the classifier. When the AUC value is [0.5, 0.7], it means that it has low accuracy, and when the value is [0.7, 0.9], it has credible accuracy, when the value is greater than 0.9, the classifier has high accuracy, when the AUC value is less than 0.5, the model has no classification significance. Fig 10 visualizes the AUC values obtained by DGCNNII and DGCNN in each epoch on the 5 test sets. The dotted line represents the AUC

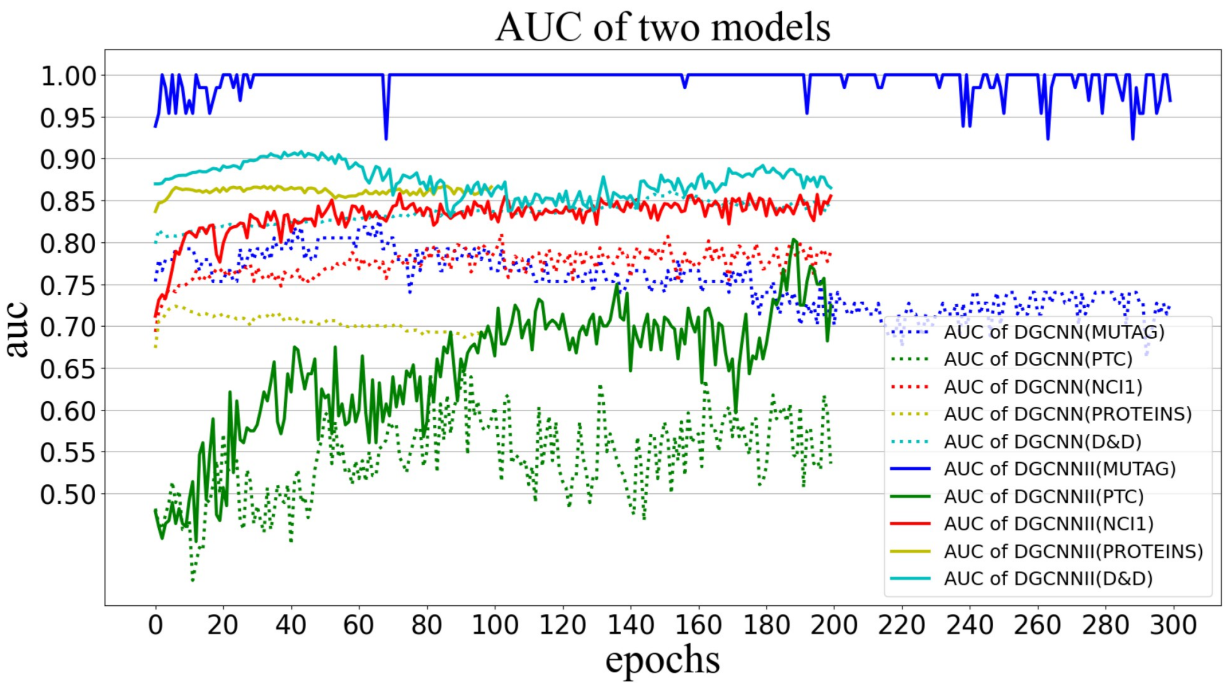

Fig 10. AUC values of DGCNNII and DGCNN on 5 test sets.

https://doi.org/10.1371/journal.pone.0279604.g010

value of the DGCNN model on the 5 datasets, the solid line represents the AUC value of the DGCNNII model, and the lines of different colors represent different datasets. MUTAG, PTC, NCI1, PROTEINS and D&D were trained for 300, 200, 200, 100, 200 epochs respectively. As can be seen from the Fig 10, the AUC values of all DGCNNII are higher than the corresponding DGCNN on each dataset. DGCNNII has high confidence with AUC values above 0.84 on all datasets except PTC, in particular, the AUC values of the MUTAG dataset basically reach 1.0. However, the AUC values of DGCNN on all datasets are lower than 0.84, and the AUC values of PTC are basically between 0.5–0.6, with relatively low confidence. But the AUC value of PTC data set on DGCNNII can reach 0.7–0.8, achieving reliable accuracy. The following Fig 11 shows the loss values of DGCNNII and DGCNN in the training process on 5 test sets. It can be seen that the loss values of DGCNNII on each data set are lower than those of DGCNN.

6 Conclusions

This paper proposes a novel graph neural network framework, named Non-local Message Passing (NLMP) framework. Compared with existing graph neural network frameworks, NLMP has many advantages. First, based on the NLMP framework, a deep graph neural network can be constructed to extract high-order neighbor nodes features almost without over-smoothing. Secondly, the neighbor aggregation scheme based on node attention weight and the graph convolution layer based on NLMP framework can extract the finer features of the nodes, highlight the key node information, and avoid the nodes over-smoothing. At the same time, various new message passing methods and attention mechanisms can be introduced based on this framework, making the model design more flexible and extensible. For the scalability of the model, the appropriate model depth can be selected by quantifying the smoothness of each layer of the DGCNNII. The ablation study in this paper also proves the effectiveness of the double-stage model structure, so we can flexibly combine and design

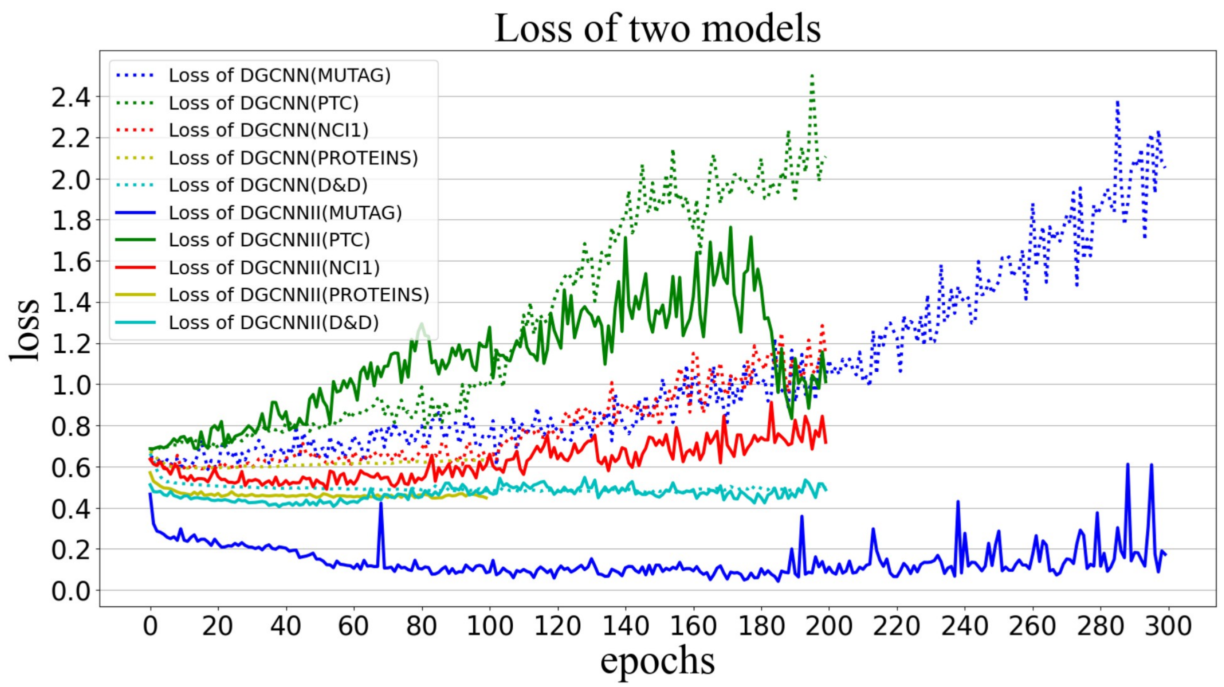

Fig 11. Loss values of DGCNNII and DGCNN on 5 test sets.

https://doi.org/10.1371/journal.pone.0279604.g011

models with different depths and layers for different data sets. The designed deep graph convolutional layer receives the original graph as input and outputs the feature matrix of the graph. Deep graph convolutional layers can be embedded into different tasks as a whole graph convolution module. For example, the graph convolution module followed by different node aggregation operations can be used for graph representation and graph classification tasks. it can also be used for link prediction tasks by extracting the subgraphs around the target link and feeding them into the graph classification model designed above. The graph convolution module can also extract node features for node prediction tasks, which makes the design of the model more flexible and scalable. Finally, the end-to-end graph classification model DGCNNII based on NLMP framework achieves better performance than a large number of baseline methods on 5 data sets. In the future, we hope to introduce a multi-head attention mechanism, and research frameworks and models that can better capture structural features, and expand the method to more datasets.

Acknowledgments

We would like to thank Hongtao Huo (People's Public Security University of China) for the insightful comments on the manuscript and his guidance and patience enlighten us not only on this paper but also our future. We gratefully acknowledge the valuable contribution by Fanliang Bu (People's Public Security University of China) during preparing the paper.

Author Contributions

Conceptualization: Yuchen Zhou.

Data curation: Yuchen Zhou, Zhiwen Hou.

Formal analysis: Yuchen Zhou, Hongtao Huo.

Investigation: Yuchen Zhou.

Methodology: Yuchen Zhou.

Project administration: Yuchen Zhou.

Resources: Yuchen Zhou, Zhiwen Hou, Fanliang Bu.

Software: Yuchen Zhou, Hongtao Huo, Zhiwen Hou.

Supervision: Hongtao Huo, Fanliang Bu.

Validation: Yuchen Zhou.

Visualization: Yuchen Zhou, Zhiwen Hou.

Writing – original draft: Yuchen Zhou.

Writing – review & editing: Yuchen Zhou, Zhiwen Hou.

References

1. LeCun Y, Bengio Y. Convolutional Networks for Images, Speech, and Time Series. The Handbook of Brain Theory and Neural Networks. Cambridge, MA, USA: MIT Press; 1998. pp. 255–258.
2. Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. CoRR. 2016; abs/1609.0. http://arxiv.org/abs/1609.02907
3. Diao Z, Wang X, Zhang D, Liu Y, Xie K, He S. Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting. The Thirty-Third {AAAI} Conference on Artificial Intelligence, {AAAI} 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, {IAAI} 2019, The Ninth {AAAI} Symposium on Educational Advances in Artificial Intelligence, {EAAI}. {AAAI} Press; 2019. pp. 890–897.
4. Qi S, Wang W, Jia B, Shen J, Zhu S-C. Learning Human-Object Interactions by Graph Parsing Neural Networks. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y, editors. Computer Vision—{ECCV} 2018–15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part {IX}. Springer; 2018. pp. 407–423.
5. Zhao L, Peng X, Tian Y, Kapadia M, Metaxas DN. Semantic Graph Convolutional Networks for 3D Human Pose Regression. {IEEE} Conference on Computer Vision and Pattern Recognition, {CVPR} 2019, Long Beach, CA, USA, June 16–20, 2019. Computer Vision Foundation / {IEEE}; 2019. pp. 3425–3435.
6. Ma J, Wen J, Zhong M, Chen W, Li X. {MMM:} Multi-source Multi-net Micro-video Recommendation with Clustered Hidden Item Representation Learning. Data Sci Eng. 2019; 4: 240–253. https://doi.org/10.1007/s41019-019-00101-4
7. Marcheggiani D, Bastings J, Titov I. Exploiting Semantics in Neural Machine Translation with Graph Convolutional Networks. In: Walker MA, Ji H, Stent A, editors. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1–6, 2018, Volume 2 (Short Papers). Association for Computational Linguistics; 2018. pp. 486–492.
8. Bastings J, Titov I, Aziz W, Marcheggiani D, Sima'an K. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In: Palmer M, Hwa R, Riedel S, editors. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, {EMNLP} 2017, Copenhagen, Denmark, September 9–11, 2017. Association for Computational Linguistics; 2017. pp. 1957–1967.
9. Li Z, Luo X, Wang B, Bertozzi AL, Xin J. A Study on Graph-Structured Recurrent Neural Networks and Sparsification with Application to Epidemic Forecasting. In: Thi HA Le, Le HM, Dinh TP, editors. Optimization of Complex Systems: Theory, Models, Algorithms and Applications, {WCGO} 2019, World Congress on Global Optimization, Metz, France, 8–10 July, 2019. Springer; 2019. pp. 730–739.
10. Singh V, Liò P. Towards Probabilistic Generative Models Harnessing Graph Neural Networks for Disease-Gene Prediction. CoRR. 2019;abs/1907.0. http://arxiv.org/abs/1907.05628
11. Li C, Goldwasser D. Encoding Social Information with Graph Convolutional Networks forPolitical Perspective Detection in News Media. In: Korhonen A, Traum DR, Màrquez L, editors. Proceedings of the 57th Conference of the Association for Computational Linguistics, {ACL} 2019, Florence, Italy, July 28-August 2, 2019, Volume 1: Long Papers. Association for Computational Linguistics; 2019. pp. 2594–2604.

12. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In: Guo Y, Farooq F, editors. Proceedings of the 24th {ACM} {SIGKDD} International Conference on Knowledge Discovery {\&} Data Mining, {KDD} 2018, London, UK, August 19–23, 2018. ACM; 2018. pp. 974–983.

13. Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. In: Fleet DJ, Pajdla T, Schiele B, Tuytelaars T, editors. Computer Vision—{ECCV} 2014—13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part {I}. Springer; 2014. pp. 818–833.

14. Li Q, Han Z, Wu X-M. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In: McIlraith SA, Weinberger KQ, editors. Proceedings of the Thirty-Second {AAAI} Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th {AAAI} Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New. {AAAI} Press; 2018. pp. 3538–3545. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098

15. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A Comprehensive Survey on Graph Neural Networks. CoRR. 2019;abs/1901.0. http://arxiv.org/abs/1901.00596

16. Zhou J, Cui G, Zhang Z, Yang C, Liu Z, Sun M. Graph Neural Networks: {A} Review of Methods and Applications. CoRR. 2018;abs/1812.0. http://arxiv.org/abs/1812.08434

17. Xu K, Hu W, Leskovec J, Jegelka S. How Powerful are Graph Neural Networks? 7th International Conference on Learning Representations, {ICLR} 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net; 2019. https://openreview.net/forum?id=ryGs6iA5Km

18. Alon U, Yahav E. On the Bottleneck of Graph Neural Networks and its Practical Implications. 9th International Conference on Learning Representations, {ICLR} 2021, Virtual Event, Austria, May 3–7, 2021. OpenReview.net; 2021. https://openreview.net/forum?id=i80OPhOCVH2

19. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. 2016 {IEEE} Conference on Computer Vision and Pattern Recognition, {CVPR} 2016, Las Vegas, NV, USA, June 27–30, 2016. {IEEE} Computer Society; 2016. pp. 770–778.

20. Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely Connected Convolutional Networks. 2017 {IEEE} Conference on Computer Vision and Pattern Recognition, {CVPR} 2017, Honolulu, HI, USA, July 21–26, 2017. {IEEE} Computer Society; 2017. pp. 2261–2269.

21. Zhang M, Cui Z, Neumann M, Chen Y. An End-to-End Deep Learning Architecture for Graph Classification. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI Press; 2018.

22. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural Message Passing for Quantum Chemistry. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning. PMLR; 2017. pp. 1263–1272. https://proceedings.mlr.press/v70/gilmer17a.html

23. Yang C, Xiao C, Ma F, Glass L, Sun J. SafeDrug: Dual Molecular Graph Encoders for Safe Drug Recommendations. CoRR. 2021;abs/2105.0. https://arxiv.org/abs/2105.02711

24. Dasoulas G, Santos L Dos, Scaman K, Virmaux A. Coloring Graph Neural Networks for Node Disambiguation. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. 2021.

25. Hamilton W, Ying Z, Leskovec J. Inductive Representation Learning on Large Graphs. In: Guyon I, Luxburg U Von, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2017. https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

26. Schlichtkrull MS, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling Relational Data with Graph Convolutional Networks. In: Gangemi A, Navigli R, Vidal M-E, Hitzler P, Troncy R, Hollink L, et al., editors. The Semantic Web—15th International Conference, {ESWC} 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings. Springer; 2018. pp. 593–607.

27. Ying Z, You J, Morris C, Ren X, Hamilton W, Leskovec J. Hierarchical Graph Representation Learning with Differentiable Pooling. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2018. https://proceedings.neurips.cc/paper/2018/file/e77dbaf6759253c7c6d0efc5690369c7-Paper.pdf

28. Wang X, Girshick R, Gupta A, He K. Non-local Neural Networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018. pp. 7794–7803.

29. Buades A, Coll B, Morel J-M. A Non-Local Algorithm for Image Denoising. 2005 {IEEE} Computer Society Conference on Computer Vision and Pattern Recognition {(CVPR} 2005), 20–26 June 2005, San Diego, CA, {USA}. {IEEE} Computer Society; 2005. pp. 60–65.

30. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, et al., editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information

Processing Systems 2017, December 4–9, 2017, Long Beach, CA, {USA}. 2017. pp. 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

31. Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph Attention Networks. 6th International Conference on Learning Representations, {ICLR} 2018, Vancouver, BC, Canada, April 30—May 3, 2018, Conference Track Proceedings. OpenReview.net; 2018. https://openreview.net/forum?id=rJXMpikCZ

32. Hoshen Y. {VAIN:} Attentional Multi-agent Predictive Modeling. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, et al., editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, {USA}. 2017. pp. 2701–2711. https://proceedings.neurips.cc/paper/2017/hash/748ba69d3e8d1af87f84fee909eef339-Abstract.html

33. Smalter AM, Huan J, Lushington GH. Graph Wavelet Alignment Kernels for Drug Virtual Screening. J Bioinform Comput Biol. 2009; 7: 473–497. https://doi.org/10.1142/s0219720009004187 PMID: 19507286

34. Mahé P, Vert J-P. Graph kernels based on tree patterns for molecules. Mach Learn. 2009; 75: 3–35. https://doi.org/10.1007/s10994-008-5086-2

35. Borgwardt KM, Kriegel H-P. Graph Kernels For Disease Outcome Prediction From Protein-Protein Interaction Networks. In: Altman RB, Dunker AK, Hunter L, Murray T, Klein TE, editors. Biocomputing 2007, Proceedings of the Pacific Symposium, Maui, Hawaii, USA, 3–7 January 2007. World Scientific; 2007. pp. 4–15. http://psb.stanford.edu/psb-online/proceedings/psb07/borgwardt.pdf

36. Fout A, Byrd J, Shariat B, Ben-Hur A. Protein Interface Prediction using Graph Convolutional Networks. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, et al., editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, {USA}. 2017. pp. 6530–6539. https://proceedings.neurips.cc/paper/2017/hash/f507783927f2ec2737ba40afbd17efb5-Abstract.html

37. Kriege NM, Johansson FD, Morris C. A survey on graph kernels. Appl Netw Sci. 2020; 5: 6. https://doi.org/10.1007/s41109-019-0195-3

38. Nikolentzos G, Siglidis G, Vazirgiannis M. Graph Kernels: {A} Survey. J Artif Intell Res. 2021; 72: 943–1027. https://doi.org/10.1613/jair.1.13225

39. Ma G, Ahmed NK, Willke TL, Yu PS. Deep graph similarity learning: a survey. Data Min Knowl Discov. 2021; 35: 688–725. https://doi.org/10.1007/s10618-020-00733-5

40. Khoshraftar S, An A. A Survey on Graph Representation Learning Methods. CoRR. 2022;abs/2204.0.

41. Ma Y, Wang S, Aggarwal CC, Tang J. Graph Convolutional Networks with EigenPooling. In: Teredesai A, Kumar V, Li Y, Rosales R, Terzi E, Karypis G, editors. Proceedings of the 25th {ACM} {SIGKDD} International Conference on Knowledge Discovery {\&} Data Mining, {KDD} 2019, Anchorage, AK, USA, August 4–8, 2019. ACM; 2019. pp. 723–731.

42. Lee J, Lee I, Kang J. Self-Attention Graph Pooling. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning, {ICML} 2019, 9–15 June 2019, Long Beach, California, {USA}. PMLR; 2019. pp. 3734–3743. http://proceedings.mlr.press/v97/lee19c.html

43. Gao H, Ji S. Graph U-Nets. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning, {ICML} 2019, 9–15 June 2019, Long Beach, California, {USA}. PMLR; 2019. pp. 2083–2092. http://proceedings.mlr.press/v97/gao19a.html

44. Klicpera J, Bojchevski A, Günnemann S. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. 7th International Conference on Learning Representations, {ICLR} 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net; 2019. https://openreview.net/forum?id=H1gL-2A9Ym

45. Rong Y, Huang W, Xu T, Huang J. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. 8th International Conference on Learning Representations, {ICLR} 2020, Addis Ababa, Ethiopia, April 26–30, 2020. OpenReview.net; 2020. https://openreview.net/forum?id=Hkx1qkrKPr

46. Feng W, Zhang J, Dong Y, Han Y, Luan H, Xu Q, et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in Neural Information Processing Systems. Curran Associates, Inc.; 2020. pp. 22092–22103. https://proceedings.neurips.cc/paper/2020/file/fb4c835feb0a65cc39739320d7a51c02-Paper.pdf

47. Wu F, Jr. AHS, Zhang T, Fifty C, Yu T, Weinberger KQ. Simplifying Graph Convolutional Networks. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning, {ICML} 2019, 9–15 June 2019, Long Beach, California, {USA}. PMLR; 2019. pp. 6861–6871. http://proceedings.mlr.press/v97/wu19e.html

48. Wang G, Ying R, Huang J, Leskovec J. Multi-hop Attention Graph Neural Networks. In: Zhou Z-H, editor. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, {IJCAI} 2021, Virtual Event / Montreal, Canada, 19–27 August 2021. ijcai.org; 2021. pp. 3089–3096.

49. Tomasi C, Manduchi R. Bilateral Filtering for Gray and Color Images. Proceedings of the Sixth International Conference on Computer Vision (ICCV-98), Bombay, India, January 4–7, 1998. {IEEE} Computer Society; 1998. pp. 839–846.

50. Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and Deep Graph Convolutional Networks. Proceedings of the 37th International Conference on Machine Learning, {ICML} 2020, 13–18 July 2020, Virtual Event. PMLR; 2020. pp. 1725–1735. http://proceedings.mlr.press/v119/chen20v.html

51. Li R, Wang S, Zhu F, Huang J. Adaptive Graph Convolutional Neural Networks. In: McIlraith SA, Weinberger KQ, editors. Proceedings of the Thirty-Second {AAAI} Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th {AAAI} Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New. {AAAI} Press; 2018. pp. 3546–3553. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16642

52. Li Y, Tarlow D, Brockschmidt M, Zemel RS. Gated Graph Sequence Neural Networks. In: Bengio Y, LeCun Y, editors. 4th International Conference on Learning Representations, {ICLR} 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings. 2016. http://arxiv.org/abs/1511.05493

53. Debnath AK, de Compadre RL, Debnath G, Shusterman AJ, 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. https://doi.org/10.1021/jm00106a046 PMID: 1995902

54. Toivonen H, Srinivasan A, King RD, Kramer S, Helma C. Statistical Evaluation of the Predictive Toxicology Challenge 2000–2001. Bioinform. 2003; 19: 1183–1193. https://doi.org/10.1093/bioinformatics/btg130 PMID: 12835260

55. Borgwardt KM, Ong CS, Schönauer S, Vishwanathan SVN, Smola AJ, Kriegel H-P. Protein function prediction via graph kernels. Proceedings Thirteenth International Conference on Intelligent Systems for Molecular Biology 2005, Detroit, MI, USA, 25–29 June 2005. 2005. pp. 47–56.

56. Dobson PD, Doig AJ. Distinguishing Enzyme Structures from Non-enzymes Without Alignments. J Mol Biol. 2003; 330: 771–783. https://doi.org/10.1016/s0022-2836(03)00628-4 PMID: 12850146

57. Wale N, Watson IA, Karypis G. Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl Inf Syst. 2008; 14: 347–375. https://doi.org/10.1007/s10115-007-0103-5

58. Kingma DP, Ba J. Adam: {A} Method for Stochastic Optimization. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, {ICLR} 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings. 2015. http://arxiv.org/abs/1412.6980

59. Borgwardt KM, Kriegel H-P. Shortest-Path Kernels on Graphs. Proceedings of the 5th {IEEE} International Conference on Data Mining {(ICDM} 2005), 27–30 November 2005, Houston, Texas, {USA}. {IEEE} Computer Society; 2005. pp. 74–81.

60. Shervashidze N, Vishwanathan SVN, Petri T, Mehlhorn K, Borgwardt KM. Efficient graphlet kernels for large graph comparison. In: Dyk DA Van, Welling M, editors. Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, {AISTATS} 2009, Clearwater Beach, Florida, USA, April 16–18, 2009. JMLR.org; 2009. pp. 488–495. http://proceedings.mlr.press/v5/shervashidze09a.html

61. Vishwanathan SVN, Schraudolph NN, Kondor R, Borgwardt KM. Graph Kernels. J Mach Learn Res. 2010; 11: 1201–1242.

62. Shervashidze N, Schweitzer P, van Leeuwen EJ, Mehlhorn K, Borgwardt KM. Weisfeiler-Lehman Graph Kernels. J Mach Learn Res. 2011; 12: 2539–2561.

63. Neumann M, Patricia N, Garnett R, Kersting K. Efficient Graph Kernels by Randomization. In: Flach PA, Bie T De, Cristianini N, editors. Machine Learning and Knowledge Discovery in Databases—European Conference, {ECML} {PKDD} 2012, Bristol, UK, September 24–28, 2012 Proceedings, Part {I}. Springer; 2012. pp. 378–393.

64. Yanardag P, Vishwanathan SVN. Deep Graph Kernels. In: Cao L, Zhang C, Joachims T, Webb GI, Margineantu DD, Williams G, editors. Proceedings of the 21th {ACM} {SIGKDD} International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10–13, 2015. ACM; 2015. pp. 1365–1374.

65. Atwood J, Towsley D. Diffusion-Convolutional Neural Networks. In: Lee DD, Sugiyama M, von Luxburg U, Guyon I, Garnett R, editors. Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5–10, 2016, Barcelona, Spain. 2016. pp. 1993–2001. https://proceedings.neurips.cc/paper/2016/hash/390e982518a50e280d8e2b535462ec1f-Abstract.html

66. Niepert M, Ahmed M, Kutzkov K. Learning Convolutional Neural Networks for Graphs. In: Balcan M-F, Weinberger KQ, editors. Proceedings of the 33nd International Conference on Machine Learning, {ICML} 2016, New York City, NY, USA, June 19–24, 2016. JMLR.org; 2016. pp. 2014–2023. http://proceedings.mlr.press/v48/niepert16.html

67. Simonovsky M, Komodakis N. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. 2017 {IEEE} Conference on Computer Vision and Pattern Recognition, {CVPR} 2017, Honolulu, HI, USA, July 21–26, 2017. {IEEE} Computer Society; 2017. pp. 29–38.

68. Verma S, Zhang Z-L. Graph Capsule Convolutional Neural Networks. CoRR. 2018;abs/1805.0. http://arxiv.org/abs/1805.08090

69. Ivanov S, Burnaev E. Anonymous Walk Embeddings. In: Dy JG, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning, {ICML} 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018. PMLR; 2018. pp. 2191–2200. http://proceedings.mlr.press/v80/ivanov18a.html

70. Taheri A. Learning Graph Representations with Recurrent Neural Network Autoencoders. 2018.

71. Yang C, Liu M, Zheng VW, Han J. Node, Motif and Subgraph: Leveraging Network Functional Blocks Through Structural Convolution. In: Brandes U, Reddy C, Tagarelli A, editors. {IEEE/ACM} 2018 International Conference on Advances in Social Networks Analysis and Mining, {ASONAM} 2018, Barcelona, Spain, August 28–31, 2018. {IEEE} Computer Society; 2018. pp. 47–52.

72. Xinyi Z, Chen L. Capsule Graph Neural Network. 7th International Conference on Learning Representations, {ICLR} 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net; 2019. https://openreview.net/forum?id=Byl8BnRcYm

73. Peng H, Li J, Gong Q, Ning Y, Wang S, He L. Motif-Matching Based Subgraph-Level Attentional Convolutional Network for Graph Classification. The Thirty-Fourth {AAAI} Conference on Artificial Intelligence, {AAAI} 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, {IAAI} 2020, The Tenth {AAAI} Symposium on Educational Advances in Artificial Intelligence, {EAAI. {AAAI} Press; 2020. pp. 5387–5394. https://ojs.aaai.org/index.php/AAAI/article/view/5987

74. Chen D, Lin Y, Li W, Li P, Zhou J, Sun X. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. The Thirty-Fourth {AAAI} Conference on Artificial Intelligence, {AAAI} 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, {IAAI} 2020, The Tenth {AAAI} Symposium on Educational Advances in Artificial Intelligence, {EAAI} 2020, New York, NY, USA, February 7–12, 2020. {AAAI} Press; 2020. pp. 3438–3445. https://ojs.aaai.org/index.php/AAAI/article/view/5747