



Evaluating mesh quality with graph neural networks

Zhichao Wang^{1,2} · Xinhai Chen^{1,2} · Tiejue Li² · Chunye Gong^{1,2} · Yufei Pang³ · Jie Liu^{1,2}

Received: 7 March 2022 / Accepted: 18 July 2022 / Published online: 30 July 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract

The quality of the finite element mesh has a considerable effect on the efficiency and accuracy of computational fluid dynamics (CFD) simulations. To ensure the generated mesh is of good quality, many quality metrics have been proposed to assess the generated mesh, such as aspect ratio, skewness, Jacobian ratio, etc. Such metrics, however, are primarily employed to detect locally distorted mesh elements. There are still no justifiable thresholds for determining whether the generated mesh is of sufficient quality for simulation. Consequently, it is necessary for the professionals to assess the generated mesh afterward which is time-consuming and labor-intensive work. With the ability to learn features on the graph, the graph neural networks have been successfully applied in many application areas to reduce human-computer interaction. In this paper, we define mesh quality evaluation as a graph classification problem. We first propose a novel and sparse-implemented algorithm to transform the mesh data into graph data. We then introduce a deep graph neural network, GMeshNet, to evaluate the mesh quality. Experimental results on the NACA-Market and NACA6510 mesh datasets demonstrate the effectiveness of our proposed network.

Keywords Computational fluid dynamics · Mesh quality evaluation · Graph neural network · Deep learning application

1 Introduction

Computational fluid dynamics (CFD) has made great progress in recent decades and has been widely applied in a variety of industries, including the aerospace industry [1], automotive engineering [2], electro-thermal simulation [3], physical simulation [4] and others. In the CFD workflow, finite mesh generation is a crucial procedure that

significantly influences simulation efficiency and accuracy [5]. Degenerate mesh elements with undesired qualities such as large or small angles or poor shapes can induce ill-conditioning of the matrix system and decrease convergence rate. Many automatic methods have been proposed to generate finite meshes. However, the meshes required for CFD simulation are closely tied to the physical problems. The meshes generated by automatic methods may not assure acceptable qualities [6]. Therefore, after the mesh generation process, the professionals usually need to check and evaluate mesh qualities, which is time-consuming and tedious work. The manual evaluation remarkably extends the CFD workflow.

Many metrics have been introduced in mesh generation to evaluate mesh quality. For example, aspect ratio, the inner angle of a triangle element, skewness, Jacobian ratio, and other metrics are applied to evaluate two-dimensional meshes. Volumetric skew and volumetric collapse are commonly used metrics to assess tetrahedral meshes in three dimensions [7]. The above metrics are widely adopted by the automatic mesh generation methods to guide the generating process. However, those quality metrics are flawed and can only be utilized to assess mesh quality in a limited way: on the one hand, most metrics are element-based, meaning that they are applied as a scalar function to a finite mesh element.

Zhichao Wang and Xinhai Chen contributed equally to this work.

✉ Jie Liu
liujie@nudt.edu.cn
Zhichao Wang
wangzhichao@nudt.edu.cn
Xinhai Chen
chenxinhai16@nudt.edu.cn

¹ Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha 410073, China

² Laboratory of Software Engineering for Complex System, National University of Defense Technology, Changsha 410073, China

³ China Aerodynamics Research and Development Center, Mianyang 621000, China

Thus, it represents local features instead of regional or global features. Their thresholds are usually based on the professionals' subjective experience. On the other hand, some metrics may have no bearing on the accuracy of computations. Knupp [8] investigates the relationship between commonly used mesh quality metrics with industry, the truncation error, and interpolation error. Their results show that some mesh quality metrics may have no connection with the errors. For these two reasons, the mesh quality metrics are primarily utilized to detect defective mesh elements and guide the automatic mesh generation and postprocessing.

With the development of graphics processing unit (GPU) and accessibility to big data, deep learning has been applied in various domains, including computational vision, natural language processing, and machine translation [9]. The deep learning method can learn an abstract function which maps the input data to some target. The method interactively optimizes the model's parameters through the gradient descent algorithm. In recent years, graph neural networks (GNNs) have shown the ability to perform deep learning on graph data. They have been successfully applied to issues involving graph data as input, such as node classification, graph classification, network embedding [10]. In this paper, we utilize GNN to evaluate the mesh quality without human involvement. We introduce a preprocessing algorithm for transforming mesh data to graph data that can be applied for meshes of different topologies. We also propose a deep graph neural network, GMeshNet, to evaluate the mesh qualities by categorizing meshes by pre-defined classes. The main contributions of our work are as follows:

1. We introduce two graph representation schemes for meshes, and a novel, sparse-implemented preprocessing algorithm to transform CFD mesh data to graph data. The algorithm is applicable to different meshes, such as structured meshes or unstructured meshes.
2. We propose a deep graph neural network, GMeshNet, for the mesh quality evaluation. The network can utilize the topology of the meshes to guide the mesh evaluation.
3. We validate GMeshNet on the real world mesh datasets NACA-Market [11] and NACA6510. It achieves high performance in detecting and recalling distorted meshes. The experiment results show the effectiveness of our proposed network.

The remainder of this paper is organized as follows. We present related work on mesh quality metrics and mesh evaluation methods with machine learning in Sect. 2. In Sect. 3, we introduce our data transform algorithm and our network. In Sect. 4, we evaluate our proposed network on the mesh benchmark datasets, NACA-Market and NACA6510. We conclude and discuss future work in Sect. 5.

2 Related work

Over the years, mesh quality evaluation has been studied in many different methods. Since it is difficult to define an evaluation function to take the entire mesh as input, mesh evaluation usually adopts element-based mesh quality metrics, which are formulated by scalar functions. Many mesh quality metrics have been proposed and widely applied in the industry for structure meshes with different topological structure. Li et al. [7] summarizes various commonly used mesh quality metrics for both two-dimensional and three-dimensional meshes. Warp angle, for example, is an angular quality metric used to evaluate the skewness of the finite element. For triangular mesh, it is defined as

$$\theta_{\text{skew}} = 90^\circ - \min(\alpha_1, \alpha_2, \alpha_3) \quad (1)$$

where $\alpha_1, \alpha_2, \alpha_3$ is the smaller crossing angle between the triangular mesh element's center lines and medians. Jacobian Ratio, defined as Eq. (2), evaluates how far an element's shape deviates from its ideal shape.

$$J_R = \frac{\|J\|_{\min}}{\|J\|_{\max}} \quad (2)$$

where $\|J\|_{\min}$ and $\|J\|_{\max}$ are the minimum and maximum Jacobian determinants of finite elements. Sarrate et al. [12] proposes a new approach for representing triangular mesh elements in a bound domain, which can graphically depict triangular meshes. Nie et al. [13] discusses several quality metrics for tetrahedral mesh, and shows the inconsistent simulation results when adopting different metrics. Kwok and Chen [14] develop a mesh quality metric for hexahedral and wedge elements based on the aspect ratio, the warping factor, and Jacobian determinants. It exhibits an explicit correlation with the solution errors. Since the lack of mathematical theory for quality metrics, Knupp [15] places the quality metrics in an algebraic framework and provides a method of constructing, classifying and evaluating mesh quality metrics. Knupp [16] specifies essential properties that a quality metric should satisfy, as well as explicit formulas for constructing different quality metrics for various mesh elements. Despite their wide applications in the industry, those quality metrics are usually employed to detect defective mesh elements. Besides this, their thresholds are often ambiguous and highly subjective. Using different mesh quality metrics may even lead to different simulation results.

Aside from those element-based mesh quality metrics, a lot of effort has been put into evaluating mesh quality with machine learning method. We divide those works into two families. One adopts traditional machine learning methods, which use feature engineering to construct input. The other utilizes the deep learning methods, which allow the network to extract needed features on its own. The support vector

machine is the most popular linear model for classification and regression problems before the rise of deep learning [17]. The algorithm creates a hyperplane in feature space and classifies data by separating them into different subspaces. Support Vector Regression (SVR) is the regression model based on the same idea. Chetouani [18] evaluates 3D mesh quality (those meshes are used in computer vision instead of CFD) with the SVR model. It combines specified mesh quality metrics and geometric attributes as input features and then employs the SVR to predict the quality scores. Sprave and Drescher [19] consider the neighborhoods of a mesh element by evaluating and aggregating the values of its neighborhoods' quality metrics. Feedforward neural networks (FNNs) and extremely randomized trees (ExtraTrees) are used to classify mesh elements. As can be seen, such methods are not end-to-end algorithms, as they take mesh quality metrics rather than mesh geometric attributes as input. Also, determining appropriate quality metrics as input features is difficult.

With the development of GPU and the availability of big data, the deep learning method can now solve many problems that were previously difficult for machines to address. By deepening and widening the network, deep neural networks (DNNs) can extract high-level features from the raw input data independently. DNNs can perform classification under diverse application environments, such as text classification, image classification, speech recognition, etc [9]. Convolutional neural networks (CNNs) [20] are commonly used in computer vision tasks such as image classification [20], target detection [21], target segmentation [22], etc. GridNet [11] and MVE-Net [23] develop CNN-based mesh quality evaluation models based on the similar data representation between two-dimensional structural meshes and the grid data. The models take the geometric attributes of mesh elements as input features. The experimental results on structured mesh show that neural networks can evaluate mesh quality in an end-to-end manner. However, such CNN-based models are challenging to apply to three-dimensional meshes or unstructured meshes. It is difficult to represent a three-dimensional mesh with a two-dimensional grid without losing information.

Graph neural networks (GNNs) have shown their ability to handle graph data structures. They have been successfully applied in many environments, including traffic prediction [24], recommender systems [25], molecular properties prediction [26] and social influence prediction [27]. Because of the enormous success of CNNs in computer vision, a lot of effort has been put into redefining convolution and pooling operation on graphs. There are two types of convolutions on graphs, spectral convolution [28] and spacial convolution [29]. The former represents node features in the spectral domain with Fourier transform on the graph. The latter aggregates the neighbors' features of the node to generate

the hidden features. In classification tasks, convolution layers are used to extract features from raw input data, whereas pooling layers are adopted to reduce input size and widen the receptive field. The node clustering methods [30] learn a cluster assignment matrix to coarsen the nodes and the graph structure. The node drop methods [31] score every node and then perform the pooling by dropping a percentage of nodes.

Taken together, we can see that traditional mesh quality metrics have limited usage. Traditional machine learning methods for evaluating mesh quality rely heavily on the manual construction of mesh features, which is not an end-to-end approach. While the CNN-based mesh quality evaluating methods have shown promising results on 2D structured meshes, it cannot be extended to other mesh types. In summary, this paper proposes a novel mesh quality evaluation network based on GNN. The proposed method can be applied to different types of meshes and be performed in an end-to-end manner.

3 Proposed method

In this paper, we define mesh evaluation as a graph classification task under supervised learning. Through a preprocessing algorithm, we transform the meshes into graph data and construct the input features. The GMeshNet takes the graph data as input and predicts the mesh quality by classifying it into pre-defined classes.

3.1 Mesh preprocessing algorithm

The input of GNNs is graph data. However, the mesh is usually stored by a set of point coordinates or mesh elements. To apply GNNs on meshes, a graph-based representation of the mesh is required. In this part, we propose two graph representation schemes for mesh data. Furthermore, we introduce an algorithm with sparse implementation to transform the mesh data into graph data. The main advantage of our algorithm is that it can be applied to meshes with different topologies.

A graph is pair of $G = (V, E)$, where $V = \{v_i | i \in N\}$ is the set of vertices (N is the number of the vertices) and $E = \{e_{ij} | e_{ij} = (v_i, v_j) (v_i, v_j) \in V^2\}$ is the set of edges. For an undirected graph, e_{ij} is identical to e_{ji} . The finite element mesh is made up of nodes and elements. In this paper, we present two schemes for representing mesh as a graph. The element-based graph treats mesh elements as vertices and the adjacencies of mesh elements as edges of the graph. The point-based graph treats mesh nodes as vertices and the adjacencies of nodes as edges of the graph. Using the graph representation, we can represent a mesh with its input feature matrix X and adjacency matrix A . The former represents the features of the vertices (mesh nodes or elements), while the

latter represents the topology of the mesh. Figure 1 depicts the two representation schemes for a structured mesh.

Next, we introduce the algorithm for constructing the input feature matrix X and the adjacency matrix A from the raw mesh file format. For the purpose of this paper, we mainly talk about the preprocessing algorithm of the structured quadrilateral mesh. It is easy to expand to other types of meshes. In this paper, we illustrate our algorithms with the file format which stores the point coordinates of each mesh element.

3.1.1 The point-based graph

The adjacencies of mesh nodes located in the same element are easily accessible. However, one node may appear in different elements which are stored separately in the mesh file. The key point is how to get the full adjacencies of one node. The main idea is to get the index of the same node in different elements, obtain adjacency and drop duplicated nodes in different elements. We construct $X_p \in \mathbb{R}^{4N \times 3}$ by stacking the coordinates of mesh elements' nodes, where N is the number of the mesh elements, and $X_{p[i,\cdot]}$ is the node i coordinates. We have duplicated nodes in different elements, meaning that for some i, j we have $X_{p[i,\cdot]} = X_{p[j,\cdot]}$. We use $A_p \in \mathbb{R}^{4N \times 4N}$ to represent the adjacency of nodes in the same element. To get the adjacency of the nodes in separate elements, a matrix A_s is defined as

$$A_s = [a_{ij}] \in \mathbb{R}^{4N \times 4N},$$

where $a_{ij} = \begin{cases} 1, & \text{if } X_{p[i,\cdot]} = X_{p[j,\cdot]} \\ 0, & \text{otherwise} \end{cases}$ (3)

Intuitively, $a_{ij} = 1$ means that node i and node j are the same node. Then the adjacencies of the nodes in different elements can be attained by

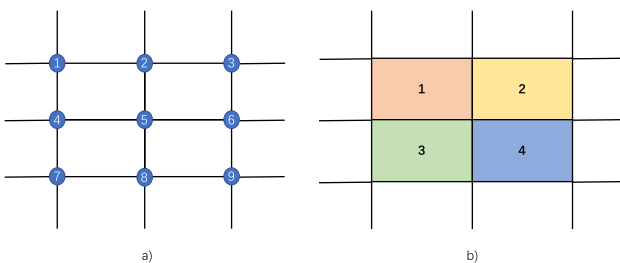


Fig. 1 Left: Mesh nodes correspond to vertices on the graph. The adjacency of vertices corresponds to the adjacency of nodes. For example, vertex 5 is adjacent to vertex 2, 4, 6 and 8. Right: Mesh elements are treated as vertices on the graph. Two elements are adjacent if and only if they share one edge. For instance, element 3 is adjacent to element 1, 4

$$A_n = A_p A_s \tag{4}$$

$$A'_n = \text{Reformat}(A_n) \tag{5}$$

The first Eq. (4) computes the adjacencies of the nodes in different elements. The Reformat operation sets the element of a matrix to one if it is greater than one. To avoid losing adjacent relationships when dropping duplicated nodes, we need to ensure that if node i is adjacent to node j , then node i is adjacent to the duplicated nodes of node j . Such operation can be done by

$$A_d = A_s A'_n, A'_d = \text{Reformat}(A_d) \tag{6}$$

The final step is dropping duplicated nodes in X_p and A'_d . Let $\text{duplicated_idx} = \{j \mid a_{ij} = 1, i < j\}$ denotes the index set of duplicated nodes to be dropped. Then the no duplicated node index is $\text{idx} = \{i \mid i \in [1, 4N], i \in \mathbb{Z}\} - \text{duplicated_idx}$. Finally, the input feature matrix X and adjacency matrix A are generated by

$$X = X_{p[\text{idx},\cdot]}, A = A'_{d[\text{idx},\text{idx}]} \tag{7}$$

3.1.2 The element-based graph

In this representation, we treat mesh elements as vertices. Two elements are adjacent if and only if the two elements share one edge, i.e., the two elements have more than one same point coordinates. The input feature matrix $X \in \mathbb{R}^{N \times d}$ is obtained directly from the raw mesh files, where N is the number of mesh elements and d is the number of element features. We still need a way to construct adjacency relations between mesh elements. Suppose that S is the element assignment matrix, which is define as $S = [s_{ij}] \in \mathbb{R}^{N_p \times N}$, where N_p is the number of the nodes, and s_{ij} equals to one if node i in element j , otherwise equals to zero. Also let $A_p \in \mathbb{R}^{N_p \times N_p}$ be the adjacency matrix of the point-based representation. Then we can obtain the adjacency matrix denoting the strength between two elements:

$$A_e = S^T A_p S \tag{8}$$

If two elements share one edge, the strength between the elements is 8. The connection strength is illustrated in Fig. 2. So, the item of final adjacency matrix A is

$$A_{[i,j]} = \begin{cases} 1, & \text{if } A_{e[i,j]} = 8 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

The above preprocessing algorithms are implemented with sparse matrixes. For a structure two-dimensional mesh with 30,400 elements, it only takes 0.79 s to

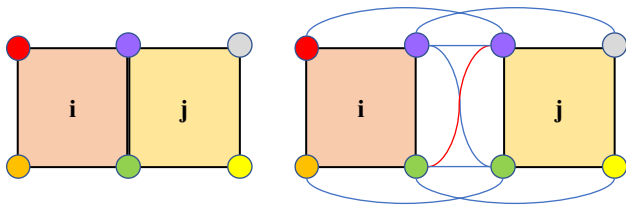


Fig. 2 Element i and j share one edge. Each element can obtain the adjacencies of the identical nodes on the other elements. The lines show the connection between the elements. Thus, in this case the connection strength of the two elements is 8

preprocess a mesh into a graph. It can be further optimized by rearranging the index order of the nodes or the elements. After obtaining X and A , Min–Max Normalization is adopted to regularize the input feature matrix.

3.2 Network architecture

Typically, a mesh may have a large number of elements. To extract high-level features from low-level features, we design a deep graph neural network. As shown in Fig. 3 our GMeshNet mainly consists of graph convolutional layers, graph pooling layers, and Jumping connections. The graph convolution layers aggregate the features of the elements on the graph layer-by-layer. And the graph pooling layer reduces the size of the graph and expands the graph convolution receptive field. Moreover, we employ Jumping connection to aggregate information under different hierarchical representations. Finally, an MLP classifies data into different classes.

3.3 Graph convolution layers

Similar to CNNs, spatial GNNs extract hidden features of vertices on the graph by aggregating the features of neighbors. Depending on the aggregation method, there are many different graph convolution operations. Since different neighbors may have different influences on the hidden features of the vertex, we adopt GAT [29] as the convolutional layer. It assigns different weights to different vertices in the neighborhood. Specifically, the inputs of the h th graph convolution are the feature matrix, $X_h = [x_1^{hT}, x_2^{hT}, \dots, x_N^{hT}] \in \mathbb{R}^{N \times m}$, and the adjacency matrix, $A_h \in \mathbb{R}^{N \times N}$, where N is the number of the vertices, m is the dim of input features, x_i^h is the features of vertex i in layer h . It then produces a new feature matrix $X'_h \in \mathbb{R}^{N \times n}$ as the input of the pooling layer, where n is the dim of output features. Suppose the weight between vertex i and vertex j is α_{ij} , then the hidden features of the vertex i are attained by

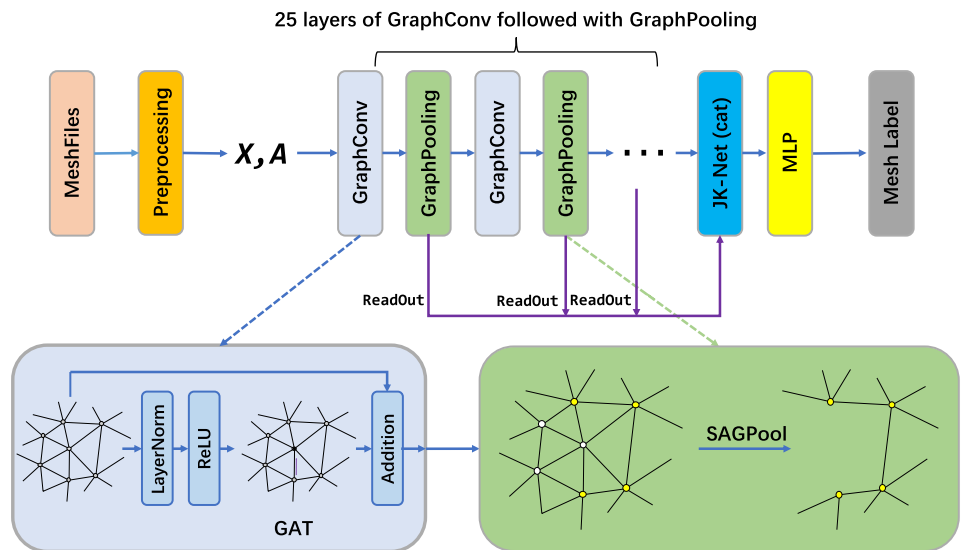
$$x_i^{h'} = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W x_j^h \right) \tag{10}$$

where σ is the ReLU function, W is the transform matrix and \mathcal{N}_i is the indexes of neighbours of vertex i . To sum up, GAT performs as

$$X'_h = GAT(X_h, A_h) \tag{11}$$

As the GNN gets deeper and deeper, it suffers from over-smoothing [32]. Over-smoothing leads to possible degradation of the performance of the network as it gets deeper and deeper. To avoid this, we adopt residual connection, which is proposed by Li et al. [33] and Li et al. [34], to tackle this problem. We compute the X_{h+1}^{res} instead of X_{h+1}

Fig. 3 The proposed network for mesh quality evaluation. Through preprocessing, the mesh is represented with the input feature matrix X and the adjacency matrix A . The network learns the features of the mesh and then predicts the mesh quality by classifying it into predefined categories. The GraphConv layer extracts features with message passing among the vertices. The Graph-Pooling layer coarsens the graph by scoring the nodes the retaining the most valuable vertices



and let the input feature dim equal to output feature dim to enable addition between residuals and original input matrix. Additionally, LayerNorm [35] and activation function σ are performed before the graph convolution. Figure 3 shows the structure of the graph convolution layers.

$$\begin{aligned} X'_h &= X_h^{\text{res}} + X_h \\ &= \text{GAT}(\sigma(\text{LayerNorm}(X_h)), A_h) + X_h \end{aligned} \quad (12)$$

3.4 Pooling layers

The mesh may consist of millions of mesh elements. By pooling the graph into smaller sizes, the network could learn local features from mesh elements and reduce its parameters and computing overhead. Due to a large number of mesh elements, it is inefficient to perform node-cluster pooling methods (e.g. DiffPool [30]) since they require to compute the assignment matrix. Therefore, we adopt SAGPool [36], implemented with sparse operations, to pool the graph data.

More specifically, the SAGPool first obtains the self-attention scores $Z \in \mathbb{R}^N$ of the nodes by Graph Convolution. Further top $[kN]$ vertices are selected based on the value of Z , where $k \in (0, 1]$ is the pooling ratio. Finally, the reconstructed feature matrix X'_h and adjacency matrix A_h are obtained by

$$\begin{aligned} idx &= \text{top} - \text{rank}(Z, [kN]) \\ Z_{\text{mask}} &= Z_{[idx]}, X'_h = X'_{h[idx,:]} \\ X_{h+1} &= X'_h \odot Z_{\text{mask}}, A_{h+1} = A_{h[idx,idx]} \end{aligned} \quad (13)$$

3.5 Jumping connection

To classify the graph, we need a representation of the entire graph. The commonly used graph representation methods are global average readout operation and global maximum readout operation. The former utilizes the average features of all vertices as graph representation, while the latter uses the maximum features of all vertices. To enhance the robustness of the algorithm, we concatenate the above two representations together. The readout operation is

$$g_h = [\text{global_avg_pool}(X_h), \text{global_max_pool}(X_h)]^T \quad (14)$$

where the two pooling operations output representations in \mathbb{R}^n .

Additionally, to consider graph representations under different levels, we apply JK-Net [37] to concatenate graph representations under different levels. The effectiveness of the JK-Net and the analysis of different connect in JK-Net are demonstrated with experiments. The final representation is then input to an MLP with SoftMax output for

classification. Between fully connected layers, BatchNorms [38] are adopted to stable the training process.

4 Experiments

In this section, we evaluate our proposed GMeshNet on the real mesh benchmark datasets, NACA-Market and NACA6510. In Sect. 4.1, we introduce the datasets and the features of the input. Section 4.2 demonstrates the training procedures. The results of the network are discussed in Sect. 4.3. Furthermore, we discuss the effect of hyper-parameters and structures of the Network on the performance in the last section.

4.1 Datasets

We evaluate GMeshNet on two mesh benchmark datasets, NACA-Market and NACA6510. NACA-Market [11] is a two-dimensional structured mesh dataset proposed for the mesh quality evaluation task. It has a total of 10240 meshes with different sizes. The distorted meshes are generated by modifying points and lines of the well-shaped mesh. Meshes are classified into eight types depending on their smoothness, orthogonality, and spacing distribution. We verify the mesh quality with CFD solver in terms of solution accuracy and convergence speed. Figure 4 shows some examples of meshes with different properties. To illustrate the generalization performance of our network under different airfoils, we used the same method to construct the NACA6510 dataset (a total of 1024 meshes) from the NACA6510 airfoil, which is shown in Fig. 4e. In experiments, we choose the element-based representation to represent mesh. We leave the point-based representation for future work because it performs poorly in the experiments. Some insights are provided in Sect. 4.4.3 to explain the difficulties of learning from point-based representation. The edge length, the maximum angle and the area of the mesh element are adopted as vertices features.

4.2 Training procedures

During our training, 10% of the meshes are used as testing data to get the final accuracy of the network. 10% of the meshes are employed for validation in training. The remaining meshes are adopted as the training data. We select AMS-Grad [39] with a initial learning rate of 1e-2 as the optimizer, and manually adjust the learning rate in the training. We use cross-entropy as the loss function and add L2 regularization with 1e-4 weight decay for the NACA-Market dataset and 5e-4 for the NACA6510 dataset. The batch size is set to 32. For all pooling layers, the pooling ratio is set to 0.78. The experiments are carried out on two RTX Titan GPUs.

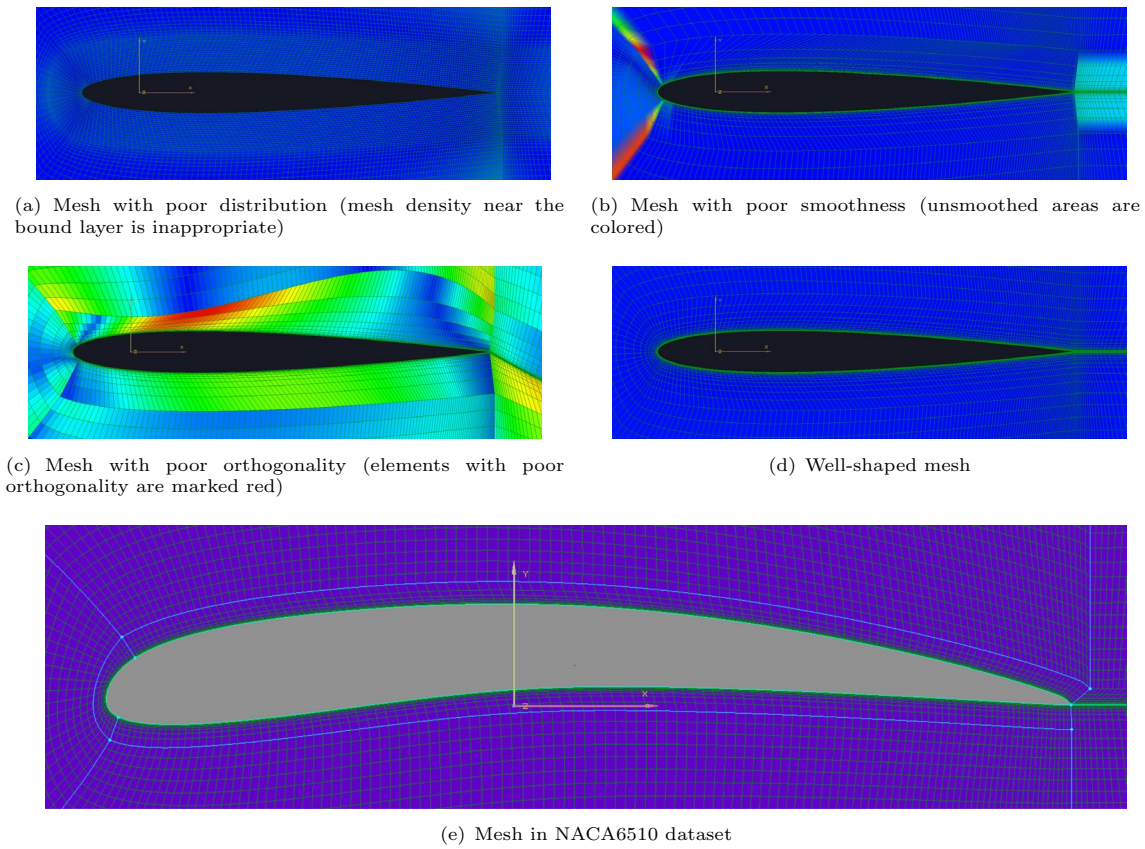


Fig. 4 Examples of meshes in NACA-Market [11] and NACA6510 dataset

Table 1 Confusion matrix of the GMeshNet on NACA-market

Labels	W (%)	N-O (%)	N-S (%)	N-D (%)	N-OS (%)	N-OD (%)	N-SD (%)	N-OSD (%)
W	99.22	0.00	0.73	0.00	0.00	0.00	0.00	0.00
N-O	0.00	94.80	0.00	0.00	4.23	0.71	0.00	0.00
N-S	13.28	0.00	87.18	0.00	0.00	0.00	0.43	0.00
N-D	0.78	0.00	0.00	97.25	0.00	0.00	2.16	0.00
N-OS	0.00	13.60	0.73	0.00	85.00	0.35	0.00	0.83
N-OD	0.00	0.40	0.00	0.00	0.00	93.97	0.00	6.67
N-SD	0.00	0.00	1.10	9.02	0.00	0.00	88.79	0.00
N-OSD	0.00	0.00	0.00	0.00	0.77	9.57	0.00	87.92

The diagonal elements of the confusion matrix represent the percentages for which the predicted label is equal to the true label for meshes with different properties. We highlight them in bold

4.3 Network evaluation results

First, we evaluated the accuracy of the network on the test set of the NACA-Market. The classification results of different types of meshes are shown in Table 1. *O*, *S*, *D* denote the orthogonality, smoothness, and distribution density of the mesh, respectively. *N-O* represents the meshes of poor orthogonality, and *N-OD* represents the meshes poor orthogonality and distribution. The *W* represents meshes with good qualities. From the experiment results, we can

see that GMeshNet achieves high accuracy on the dataset. For mesh with good qualities, the network achieves the highest accuracy with 99.22%. The network can easily classify the meshes with poor orthogonality and distribution. For different types of meshes, the accuracy rate is all above 85%, which demonstrates the effectiveness of GMeshNet for mesh quality evaluation.

Second, we utilize recall on NACA-Market test set to evaluate the ability of GMeshNet to identify defects of the mesh. In the experiment, we calculate the recall based on

Table 2 Recall and accuracy of different networks

Mesh property	Network	
	GMeshNet (%)	GridNet (%)
Orthogonality	99.81	96.95
Smoothing	88.16	93.46
Distribution	99.21	99.92
Overall accuracy	91.76	90.65

Table 3 Recall and accuracy of GMeshNet on the NACA6510 dataset

Mesh property	Recall (%)
Orthogonality	80.73
Smoothing	78.57
Distribution	82.69
Overall accuracy	85.37

the orthogonality, smoothness and distribution of the mesh. Recall indicates the percentage of bad meshes identified by the network out of all bad meshes. For example, the recall of the *N-O* meshes is defined as

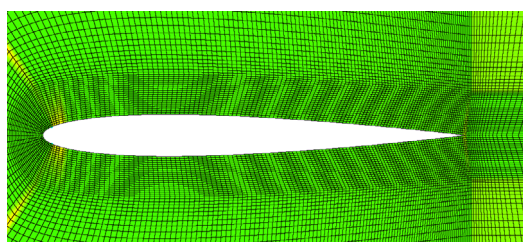
$$\text{recall}_{N-O} = \frac{\text{N-O meshes identified by the network}}{\text{all N-O meshes}} \quad (15)$$

Table 2 shows the results of recalling a different kind of defects. From Table 2, we can see that GMeshNet can

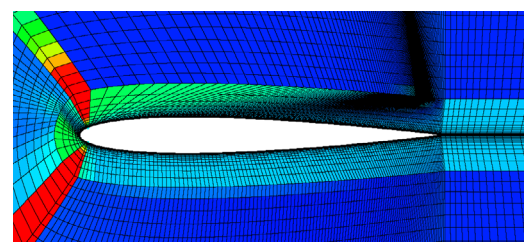
retrieve almost all bad meshes. Meshes of poor orthogonality and poor distribution can be easily distinguished with the accuracy of 99.81% and 99.21%, respectively. In comparison with GridNet [11], GMeshNet has comparable performance in terms of recall. As for the overall accuracy, GMeshNet has 1.11% improvement. But more importantly, GMeshNet has the ability to train on all types of meshes rather than structure meshes.

Third, we also validate our network by comparing it with the widely used CFD-preprocessing software ICEM CFD, which utilizes traditional metrics to evaluate mesh qualities. As shown in Fig. 5, we can see that the proposed method can correctly discriminate meshes with poor distribution, poor smoothness or poor orthogonality. Furthermore, the network is superior to traditional quality metrics. Figure 5d depicts a defective mesh which has poor-distribution. However, the ICEM CFD misclassifies it as a well-shaped mesh. In contrast, our proposed network successfully detects it, which shows the effectiveness of the proposed network compared with traditional metrics.

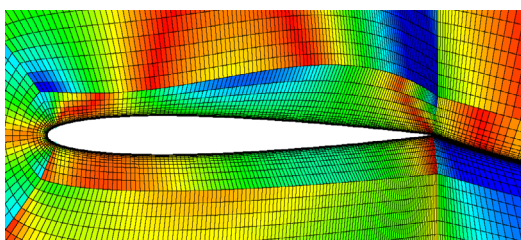
Fourth, to illustrate the generalization performance of the network, we validate the network on a different airfoil mesh dataset, NACA6510. The experimental results are shown in Table 3. We can see that the network can also achieve satisfying accuracy and recall when the airfoil changes, which shows the ability of the network to learn patterns of meshes with different airfoils. We can also observe that due to overfitting on the small-scale NACA6510 dataset, the performance is slightly inferior to that on NACA-Market.



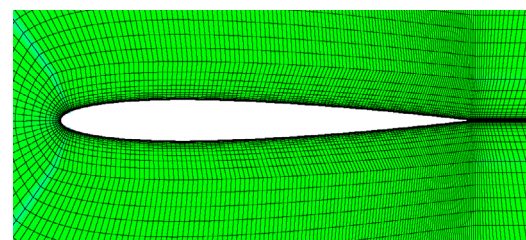
(a) Mesh with poor distribution



(b) Mesh with poor smoothness



(c) Mesh with poor orthogonality



(d) Mesh classified as well-shaped mesh by the mesh metrics and actually defective

Fig. 5 Visualization of discrimination results on NACA-Market with ICEM. Figure 5a–c, shows that our network can discriminate the defective meshes in agreement with the traditional metrics. However,

for some meshes (such as mesh in Fig. 5d) that are actually flawed, our network can correctly classify them but the traditional mesh quality evaluation metrics fail

Table 4 Test accuracy for different hyper-parameters

Network depth	0.3 (%)	0.5 (%)	0.7 (%)	0.9 (%)	Avg. (%)
Pooling ratio					
4	81.22	85.12	81.46	82.68	82.62
8	84.88	84.10	88.29	82.68	84.99
16	81.71	88.05	85.61	87.07	85.61
25	86.10	84.63	84.15	85.61	85.12
32	85.61	84.15	84.39	85.85	85.00
Avg.	83.90	85.21	84.78	84.78	84.67

In summary, our proposed GMeshNet can evaluate the mesh quality with both high accuracy and recall, has a better discriminative ability compared to traditional metrics and can be applied to different meshes.

4.4 Analysis of hyper-parameters

In this part, we discuss the effect of hyper-parameters on the performance. For the mesh evaluation tasks, selecting the appropriate hyper-parameters is beneficial to the network performance, and can reduce the computational and storage overhead. We only discuss the hyper-parameters that have a large impact on the network performance. Due to the long training time of the model on the whole dataset, we perform hyper-parameter experiments on a subset (1024 meshes of the same size) of the dataset. In the experiments, we fix other parameters and change the depth of the network and the pooling ratio of the pooling layers. The experimental results are shown in Table 4.

4.4.1 Depth of the network

As the number of network layers deepens, the performance of the network improves to some extent. The average accuracy of the 16-layer network has a 2.99% improvement than the average accuracy of the 4-layer network. However, A deeper network is not necessarily better than a shallower network. We can see that the 8-layer networks with 0.7 pooling ratio achieve the best performance. A shallow network with a suitable pooling rate may be better than a deep network with an inappropriate pooling rate. In addition, a appropriate number of network layers can also reduce the sensitivity of the network to the pooling rate. We can see that the performance of the network is less sensitive to the pooling rate when the network depth is 25 or 32.

4.4.2 Pooling ratio

The other important hyper-parameter is the pooling ratio k . Intuitively, the pooling ratio determines what percentage of vertices are left after pooling. A larger k helps to preserve

the information of the graph, while a smaller k helps to quickly reduce the size of the graph resulting to a smaller network depth. We can see that the optimal pooling ratio is related to the network depth. For 4-layer or 16-layer networks, the performance is better at the pooling ratio of 0.5. While for 8-layer networks, the appropriate pooling ratios is 0.7. 25-layer networks and 32-layer networks achieve best result at a pooling ratio of 0.3 and 0.9. On average, 0.5 is an appropriate pooling rate for networks of different depths. This experience can help us to set the pooling ratio at the beginning of constructing the mesh evaluation network.

4.4.3 Representation scheme

In Sect. 3.1 we propose two different representations, but in our experiments, we only used the element-based representation. In fact, we experimented with both representations, but the point-based representation fails to achieve satisfactory results even for simple discrimination tasks (such as train on 256 samples). We believe that the reason for this is that the point-based representation loses the information of the mesh elements. The mesh quality depends in large part on the shape of the mesh elements. However, in the point-based scheme mesh nodes become the basic units for feature learning. Because of the stochastic of neural networks, we cannot constrain how the network learns, for example, let it learn the norm of the vector which is obtained by subtracting the coordinates of two points. Thus, the graph convolution layers may not capture the edge features. However, we believe that the information about the element edges of meshes is important for network to determine mesh quality. Future work includes the use of convolution layers utilizing edge features, so that point-based representations can also achieve satisfactory results.

4.5 The analysis of network structure

In this section, we illustrate the effectiveness of JK-Net and the effect of the graph convolution layer on the network performance. The purpose of JK-Net is to consider the graph representation under different hierarchies in the network. In JK-Net, we can use different schemes to aggregate graph representations, such as as concatenation and max pooling. We test the effect of JK-Net on the performance in different aggregation schemes, including without JK-Net, with max pooling scheme and with concatenation scheme. The results are shown in Table 5. We can see that not using JK-Net the network has a large degradation in performance. Compared with the max pooling scheme, the concatenation scheme can achieve higher accuracy.

In addition to JK-Net, we also compare the effect of adopting different graph convolutions on the network performance. We select classical graph convolutions for the

Table 5 Experimental results of the JK-Net aggregation schemes

JK-Net	Accuracy (%)
None	55.37
Max pooling	77.89
Concatenation	86.34

None means that JK-Net is not used

Table 6 Experimental results of the different GraphConv

GraphConv	Train accuracy (%)	Test accuracy (%)
GCN	91.33	85.99
GraphSAGE	93.16	85.87
GAT	94.87	85.47

experiment, including GCN [28], GraphSAGE [40], GAT [29]. The experimental results are shown in Table 6. We can see that the network based on GAT achieves better train and test accuracy, which indicates that using that type of convolutions makes the network have better learning ability. However, the choice of the graph convolution layer has less impact on the performance of the network. In summary, we recommend GAT as the graph convolution for feature learning on the mesh.

5 Conclusion

The mesh quality is of great importance for the accuracy of CFD simulations. However, the widely used mesh quality metrics based on mesh elements have limited usage. The mesh quality evaluation still relies on manual work. To tackle this problem, we introduce graph neural network into the mesh quality evaluation. Firstly, we define two graph-based representations for meshes, and propose algorithms with sparse-implementation to transform any type of mesh into the graph. Secondly, a deep graph neural network, GMeshNet, is designed for mesh quality evaluation. We evaluate our network on NACA-Market and NACA0012 mesh datasets. The experimental results show the feasibility of the GNN-based network for mesh evaluation tasks for different meshes.

Future work includes extending the classification of the whole mesh to the classification of mesh elements. However, performances for the meshes with different working conditions (such as when Reynolds number and angle of attack change) and topological structures still need to be investigated in future work. By considering the flow field features and working condition features, we believe that GNN can deal with different meshes as we have shown when the

airfoil changes. Moreover, we will focus on applying the GNN-based networks to more fields in CFD, including mesh generation, mesh optimization, and others.

Author Contributions All authors contributed to the study conception and design. Data collection and analysis, network design, experiment and result analysis were performed by ZW and XC. The first draft of the manuscript was written by ZW and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This research work was supported in part by the National Numerical Windtunnel project (NNW2019ZT5-A10) and National Key Research and Development Program of China.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

- Spalart P, Venkatakrisnan V (2016) On the role and challenges of CFD in the aerospace industry. *Aeronaut J* 120(1223):209–232. <https://doi.org/10.1017/aer.2015.10>
- Watanabe N, Miyamoto S, Kuba M et al (2003) The CFD application for efficient designing in the automotive engineering. *SAE Trans* 1476–1482. <https://doi.org/10.4271/2003-01-1335>
- Das S, Paul S, Doloi B (2019) Application of cfd and vapor bubble dynamics for an efficient electro-thermal simulation of edm: an integrated approach. *Int J Adv Manuf Technol* 102. <https://doi.org/10.1007/s00170-018-3144-x>
- Wan S, Ang Y, Sato T et al (2013) Process modeling and cfd simulation of two-way abrasive flow machining. *Int J Adv Manuf Technol* 71:1077–1086. <https://doi.org/10.1007/s00170-013-5550-4>
- Ho-Le K (1988) Finite element mesh generation methods: a review and classification. *Comput Aided Des* 20(1):27–38. [https://doi.org/10.1016/0010-4485\(88\)90138-8](https://doi.org/10.1016/0010-4485(88)90138-8)
- Gammon M (2018) A review of common geometry issues affecting mesh generation. In: 2018 AIAA Aerospace Sciences Meeting, p 1402. <https://doi.org/10.2514/6.2018-1402>
- Li H et al (2012) Finite element mesh generation and decision criteria of mesh quality. *China Mech Eng* 23(3):368
- Knupp P (2007) Remarks on mesh quality. Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States)
- Guo Y, Liu Y, Oerlemans A et al (2016) Deep learning for visual understanding: a review. *Neurocomputing* 187:27–48. <https://doi.org/10.1016/j.neucom.2015.09.116>
- Wu Z, Pan S, Chen F et al (2020) A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 32(1):4–24. <https://doi.org/10.1109/TNNLS.2020.2978386>
- Chen X, Liu J, Pang Y et al (2020) Developing a new mesh quality evaluation method based on convolutional neural network. *Eng Appl Comput Fluid Mech* 14(1):391–400. <https://doi.org/10.1080/19942060.2020.1720820>
- Sarrate J, Palau J, Huerta A (2003) Numerical representation of the quality measures of triangles and triangular meshes. *Commun Numer Methods Eng* 19(7):551–561. <https://doi.org/10.1002/cnm.585>

13. Nie C, Liu J, Sun S (2003) Study on quality measures for tetrahedral mesh. *Chin J Comput Mech* 20(5):579–582
14. Kwok W, Chen Z (2000) A simple and effective mesh quality metric for hexahedral and wedge elements. In: *IMR*, pp 325–333
15. Knupp PM (2001) Algebraic mesh quality metrics. *SIAM J Sci Comput* 23(1):193–218. <https://doi.org/10.1137/S1064827500371499>
16. Knupp PM (2003) Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elem Anal Des* 39(3):217–241. [https://doi.org/10.1016/S0168-874X\(02\)00070-7](https://doi.org/10.1016/S0168-874X(02)00070-7)
17. Chauhan VK, Dahiya K, Sharma A (2019) Problem formulations and solvers in linear SVM: a review. *Artif Intell Rev* 52(2):803–855. <https://doi.org/10.1007/s10462-018-9614-6>
18. Chetouani A (2017) A 3D mesh quality metric based on features fusion. *Electron Imaging* 2017(20):4–8. <https://doi.org/10.2352/ISSN.2470-1173.2017.20.3DIPM-001>
19. Sprave J, Drescher C (2021) Evaluating the quality of finite element meshes with machine learning. [arXiv:2107.10507](https://arxiv.org/abs/2107.10507)
20. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst* 25:1097–1105. <https://doi.org/10.1145/3065386>
21. Wang J, Zheng T, Lei P et al (2019) A hierarchical convolution neural network CNN-based ship target detection method in spaceborne sar imagery. *Remote Sens* 11(6):620. <https://doi.org/10.3390/rs11060620>
22. Jalilian E, Uhl A, Kwitt R (2017) Domain adaptation for cnn based iris segmentation. In: 2017 International Conference of the Biometrics Special Interest Group (BIOSIG), pp 1–6. <https://doi.org/10.23919/BIOSIG.2017.8053502>
23. Chen X, Liu J, Gong C et al (2021) MVE-Net: An automatic 3-d structured mesh validity evaluation framework using deep neural networks. *Comput Aided Des* 141(103):104. <https://doi.org/10.1016/j.cad.2021.103104>
24. Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp 3634–3640. <https://doi.org/10.24963/ijcai.2018/505>
25. Monti F, Bronstein MM, Bresson X (2017) Geometric matrix completion with recurrent multi-graph neural networks. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp 3700–3710
26. Gilmer J, Schoenholz SS, Riley PF et al (2017) Neural message passing for quantum chemistry. In: *International conference on machine learning*, PMLR, pp 1263–1272
27. Qiu J, Tang J, Ma H et al (2018) Deepinf: Social influence prediction with deep learning. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp 2110–2119. <https://doi.org/10.1145/3219819.3220077>
28. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. <https://doi.org/10.1145/3459637.3482477>. [arXiv preprint arXiv:1609.02907](https://arxiv.org/abs/1609.02907)
29. Veličković P, Cucurull G, Casanova A et al (2017) Graph attention networks. [arXiv preprint arXiv:1710.10903](https://arxiv.org/abs/1710.10903)
30. Ying R, You J, Morris C et al (2018) Hierarchical graph representation learning with differentiable pooling. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp 4805–4815
31. Cangea C, Veličković P, Jovanović N, et al (2018) Towards sparse hierarchical graph classifiers. [arXiv preprint arXiv:1811.01287](https://arxiv.org/abs/1811.01287)
32. Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: *Thirty-Second AAAI conference on artificial intelligence*
33. Li G, Muller M, Thabet A, et al (2019) DeepGCNs: Can GCNs go as deep as CNNs? In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp 9267–9276. <https://doi.org/10.1109/ICCV.2019.00936>
34. Li G, Xiong C, Thabet A et al (2020) Deepergcn: All you need to train deeper gcns. [arXiv:2006.07739](https://arxiv.org/abs/2006.07739)
35. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. [arXiv:1607.06450](https://arxiv.org/abs/1607.06450)
36. Lee J, Lee I, Kang J (2019) Self-attention graph pooling. In: *International Conference on Machine Learning*, PMLR, pp 3734–3743
37. Xu K, Li C, Tian Y et al (2018) Representation learning on graphs with jumping knowledge networks. In: *International Conference on Machine Learning*, PMLR, pp 5453–5462
38. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML 15*, pp 448–456
39. Reddi SJ, Kale S, Kumar S (2019) On the convergence of adam and beyond. [arXiv:1904.09237](https://arxiv.org/abs/1904.09237)
40. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, NIPS'17, pp 1025–1035

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.