Applied Soft Computing Journal 187 (2026) 114221

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Neighbour-level message interaction encoding for improved representation learning on graphs

Haimin Zhang, Min Xu*

School of Electrical and Data Engineering, Faculty of Engineering and IT, University of Technology Sydney, 15 Broadway, Ultimo, NSW 2007, Australia

HIGHLIGHTS

- This paper introduces message interaction encoding for learning over graphs.
- This method addresses the issue of information loss in node embeddings.
- It is a general method that can be integrated into different graph neural networks.
- It improves the performance on various datasets across different graph tasks.

ARTICLE INFO

Keywords:
Graph convolutional networks
Graph representation learning
Neighbour-level message interaction encoding

ABSTRACT

Message passing has become the dominant framework in graph representation learning. The essential idea of the message-passing framework is to update node embeddings based on the information aggregated from local neighbours. However, most existing aggregation methods have not encoded neighbour-level message interactions into the aggregated message, resulting in information loss in embedding generation. This information loss could accumulate and become more serious as more layers are added to the graph network model. To address this issue, we propose a neighbour-level message interaction information encoding method for improving graph representation learning. For messages that are aggregated at a node, we explicitly generate an encoding between each message and the rest of the messages using an encoding function. Then we aggregate these learned encodings and take the sum of the aggregated encoding and the aggregated message to update the embedding for the node. In this way, neighbour-level message interaction information is integrated into the generated node embeddings. The proposed encoding method is a generic method that can be integrated into message-passing graph convolutional networks. Extensive experiments are conducted on six popular benchmark datasets across four highly-demanded tasks. The results show that integrating neighbour-level message interactions achieves improved performance of the base models, advancing the state of the art results for representation learning over graphs.

1. Introduction

Graph-structured data are very commonly seen throughout various domains such as neuroscience and social science. Social networks, brain functional networks, communication networks and molecular structures—all of these types of data have an underlying graph structure. Therefore, it is of considerable significance to develop models that can learn and generalize from these graph-structured data. Over the past years, increasing studies have been devoted to representation learning on graphs, including generalizations of convolutional neural networks to non-Euclidean data, techniques for geometry processing, and neural message passing approaches [1]. These efforts have produced new influential results in a variety of domains, including recommendation systems [2], social network analysis [3,4], time series analysis [5] and various vision tasks [6,7].

Basically, a graph contains a set of nodes and a set of edges between pairs of these nodes [1]. For instance, in a social network nodes could be used to represent users and edges can be used to represent a connection between pairs of users. Unlike images and natural languages, which essentially have a grid or sequence structure, graph-structured data have a fundamental structure that exists in a non-Euclidean space. It is a complicated task to develop models that are able to learn and

* Corresponding author.
Email address: min.xu@uts.edu.au (M. Xu).

https://doi.org/10.1016/j.asoc.2025.114221
Received 12 January 2025; Received in revised form 25 July 2025; Accepted 31 October 2025
Available online 14 November 2025

generalize over these graph-structured data. Early attempts employed recursive neural networks to process directed acyclic-structured graph data. Later on, researchers developed models that can deal with general graphs [8,9]. These early models primarily contain a recurrent process that iteratively updates node states and exchanges information between nodes until these node states reach a stable equilibrium.

Bruna [10] first derived graph convolutional network model that extends the concept of convolution in signal processing to non-Euclidean graphs. Since then, a variety of graph neural network models have been developed, especially in recent years. While these graph neural network models can be motivated in different ways, they can be grouped into spectral approaches and spatial approaches. The spectral approaches, which generalize the notion of signals and convolutions to the graph domain, define convolutions through an extension of the Fourier transform to graphs [10]. Unlike the spectral approaches, the spatial approaches directly define convolutions on spatially localized neighbours and generate features for nodes according to information aggregated from local neighbourhood [11,12]. Both spectral and spatial graph convolutional networks are essentially neural networks [13] that utilize a message passing paradigm. In this paradigm, messages are exchanged between nodes and node representations are updated based on the messages using neural networks.

Message passing is significant for current graph convolutional networks. The idea of message passing is to generate embeddings for every node through aggregating information from a local neighbourhood. The most commonly used aggregation operation simply takes the sum of the messages from a node's local neighbourhood. For instance, in the GCN model [14], features from a node's local neighbours are multiplied by a factor equal to the reciprocal of the node's neighbour set size and then added together to update the node's representation. In GatedGCN [15], features from a node's local neighbours are multiplied by an edge gate vector computed by the gating mechanism and then the gated representations are added together to update the node's embedding.

Message aggregation is a key step in the message passing framework. However, in most existing aggregation methods the neighbour-level message interaction information is not encoded in the aggregated message. Therefore, there exists information loss at each iteration of message passing, and this information loss could accumulate as more layers are added to the model, resulting in reduced performance in representation learning over graphs. To deal with this issue, we propose a neighbour-level message interaction encoding method for improving graph convolutional networks. An illustration of our method is demonstrated in Fig. 1. During each round of message passing, we first follow the conventional approach to aggregate messages from a node's local neighbourhood to generate an aggregated message for the node. Then for each message from a neighbour node, we explicitly learn an encoding between this message and the aggregated message from the rest of the neighbour nodes in the local neighbourhood. We aggregate these learned encodings and combine the aggregated encoding with the aggregated message to generate an updated embedding for the center node of the local neighbourhood. Because each learned encoding encodes the interaction information between a node and the rest of the nodes, the neighbour-level message interaction information is integrated into the updated embeddings.

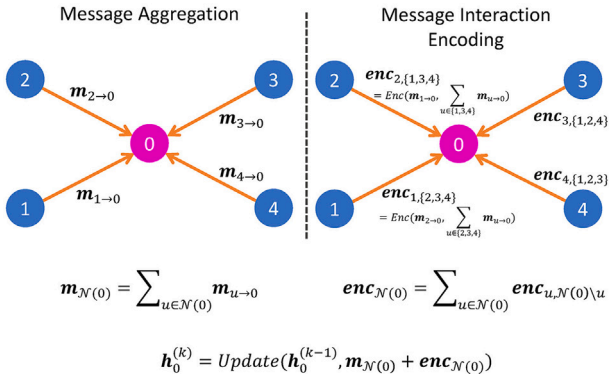

Fig. 1. For each message that passes from a neighbour node to the target node, an encoding is learned between this message and the aggregated message from the rest of the neighbours. These learned encodings are aggregated to generate a neighbour-level message interaction encoding. The sum of the aggregated message and neighbour-level message interaction encoding is taken to update the embedding for the target node.

The proposed neighbour-level message encoding method is a generic approach that can be incorporated into message-passing graph convolutional networks. We validate the proposed method on two base models: GCN and GatedGCN. We carry out experiments on six benchmark datasets, including MNIST, CIFAR10, PATTERN, CLUSTER, TSP and ZINC [16]. These experiments are carried out on four graph domain tasks, including superpixel graph classification, node classification, edge prediction and graph regression. The experimental results show that integrating neighbour-level message interaction information consistently improves the performance of the base models, advancing the state-of-the-art performance for representation learning on these datasets.

To summarize, this paper provides the following contributions.

- This paper proposes a neighbour-level message interaction encoding method for improving graph convolutional networks. For each message that passes from a neighbour node to the target node, we learn an encoding between this message and the aggregated message from the rest of the neighbours. We aggregate these learned encodings to generate a neighbour-level message encoding and take the sum of the aggregated message and the neighbour-level message encoding to update the target node's embedding. This significantly improves the representational ability of generated node embeddings.
- The proposed encoding method is a generic method that can be integrated into current message passing graph convolutional networks. It explicitly addresses the issue of information loss in the embedding generation process, which exists in most existing graph convolutional networks.
- We demonstrate that the proposed method consistently improves the performance of the base models on six benchmark datasets across four domain tasks, advancing the state of the art performance for representation learning on these benchmark datasets.

2. Related work

Generalizing neural networks to non-Euclidean data has been studied for decades. The first-generation graph neural network models were developed by Gori et al. [8] and Scarselli et al. [9]. These early models primarily use recurrent neural networks to generate node representations in an iterative manner. This process is usually computationally expensive. Driven by convolutional neural networks' considerable success, the past years have seen a surge in studies on generalizing convolutional neural networks to the graph domain. These efforts have resulted in new theories and a variety of models for representation learning on graphs.

Most existing graph convolutional networks can be categorized into spectral approaches and spatial approaches [17]. The spectral approaches are motivated by the spectral graph theory. The key feature of the spectral approaches is that they define graph convolutions through an extension of the Fourier transform to graphs. Bruna et al. [10] developed the basic graph convolutional network model, in which convolutions are formulated in the Fourier domain based on the eigendecomposition of the graph Laplacian matrix. This basic graph convolutional network model was later improved by the Chebyshev graph network (ChebNet) model [18]. The ChebNet model constructs convolutions according to the Chebyshev expansion of the graph Laplacian, which eliminates the process for graph Laplacian decomposition and can result in spatially localized kernels. Finally, the influential GCN model

[14] was introduced, addressing the limitations in the previous methods. The GCN model is a layered architecture based on a first-order approximation of spectral convolutions on graphs.

Unlike the spectral approaches, the spatial approaches directly define convolutions on graph nodes in a local neighbourhood and generate node representations by aggregating information from a local neighbourhood. Monti et al. [19] proposed the mixture model network (MoNet) model, which is a spatial approach that generalizes convolutional neural networks to graphs and manifolds. Hamilton et al. [11] developed the GraphSAGE model, which generates representations for nodes by sampling a fixed-size set of neighbours and aggregating features from local neighbours using an aggregation function. Velickovic et al. [12] introduced the integration of the self-attention mechanism which learns a weight for each feature from a neighbour node in the aggregation step. This method enables the model to focus on important information in local neighbourhood feature aggregation. Lee et al. [20] introduced the integration of edge attention and hop attention for embedding generation and showed that this method helps to alleviate the issue of over-smoothed features and attentions. Gao et al. [21] proposed segmenting a graph into patches using spectral clustering. Then a graph neural network is used to learn patch-level representations and a Transformer model is used to generate graph-level representations. Compared with the spectral approaches, the spatial approaches are much more efficient and can be scalable to arbitrarily structured graphs.

In recent years, there has been a growing endeavor to develop deeper graph network models. Chen et al. [22] introduced the GCNII model, which extends the GCN model with initial residual connection and identity mapping. They theoretically showed that a GCNII model with K layers is able to represent a spectral filter in polynomial form up to order K with arbitrary coefficients, and this property makes it possible to build deep graph convolutional network models. Zhao et al. [23] introduced a normalization layer called PairNorm, which maintains a consistent total pairwise feature distance across layers, in order to address the over-smoothing issue in building deeper graph network models. Zhang et al. [24] proposed a method that stochastically scales features and gradients during training. They showed that this method helps to improve the learning performance and generalization performance. Zhu et al. [25] studied the impact of inter-domain node interactions and introduced a propagation structure-aware graph Transformer model for fake news detection. Gao et al. [26] investigated how the interactions among closely connected neighbours affect prediction and classification in dynamic graphs.

3. Methodology

In this section, we first describe the notations and the message passing framework. Then we introduce the proposed neighbour-level message interaction encoding method for improving message-passing graph convolutional networks.

3.1. Notations

Formally, a graph $G = (V, E)$ is defined through a set of nodes, also called vertices, V and a set of edges E between pairs of these nodes. An edge from node $u \in V$ to node $v \in V$ is represented as (u, v). $\mathcal{N}(u)$ denotes the set of node u's neighbouring nodes. The adjacency matrix of the graph G is denoted as $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$, in which $\mathbf{A}_{u,v} = 1$ if $(u, v) \in E$ or $\mathbf{A}_{u,v} = 0$ otherwise. The degree matrix $\mathbf{D}$ of G is a $|V| \times |V|$ diagonal matrix wherein $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The node-level feature or attribute associated with each node $u \in V$ is denoted as $\mathbf{x}_u$.

The Laplacian matrix of G is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, and the symmetric normalized Laplacian is defined as $\mathbf{L}^{sym} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is an identity matrix. The symmetric normalized Laplacian $\mathbf{L}^{sym}$ is positive semidefinite and can be factorized as $\mathbf{L}^{sym} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_{|V|})$ is a diagonal matrix of the eigenvalues and $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_{|V|}] \in \mathbb{R}^{|V|\times|V|}$ is a matrix of eigenvectors.

3.2. The message-passing framework

Graph convolutional network models can be motivated in different ways. From one perspective, the basic graph convolutional network model is derived based on the spectral graph theory, as a generalization of Euclidean convolutions to non-Euclidean graphs [10]. The graph convolution is defined as the product of a signal $\mathbf{s} \in \mathbb{R}^N$ with a filter g_θ parameterized by $\theta \in \mathbb{R}^N$ in the Fourier domain:

$$g_\theta * \mathbf{s} = \mathbf{U} g^*_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{s}, \tag{1}$$

where $*$ denotes the convolution operation. g_θ can be understood as a function of the eigenvalues, i.e., $g_\theta = g^*_\theta(\mathbf{\Lambda})$, and $\mathbf{U}^T\mathbf{s}$ is the graph Fourier transform of $\mathbf{s}$.

From another perspective, graph convolutions can be defined on a spatially localized neighbourhood. The representations for nodes are generated by aggregating information from a local neighbourhood. This behaviour is analogous to that of the convolutional kernels in convolutional neural networks, which aggregate features from spatially defined patches in an image. Fundamentally, spectral and spatial graph convolutional networks are neural networks [13] that use a message-passing form. In this form, messages are exchanged between graph nodes and node representations are updated based on the messages using neural networks [27]. At each layer of message passing, a hidden representation $\mathbf{h}^{(k)}_u$ for each node $u \in V$ is updated according to the information aggregated from the node's local neighbourhood. According to Hamilton [1], this message passing framework can be expressed as follows:

$$\mathbf{h}^{(k)}_u = Update^{(k)}(\mathbf{h}^{(k-1)}_u, Agg^{(k)}(\mathbf{h}^{(k-1)}_v, \forall v \in \mathcal{N}(u))), \tag{2}$$

where $Update$ and Agg are a differentiable functions, e.g., neural networks. The superscripts are used for differentiating the embeddings and functions at different layers of message passing. The embeddings at $k = 0$ are initialized to the node-level features, i.e., $\mathbf{h}^{(0)}_u = \mathbf{x}_u, \forall u \in V$. During each iteration of message passing, the Agg function aggregates the embeddings of nodes in u's graph neighbourhood $\mathcal{N}(u)$ and generates a representation according to the aggregated neighbourhood information. The $Update$ function then generates an updated embedding for node u based on the aggregated information and its previous layer embedding. After k message passing iterations, the embedding for each node contains information from its k-hop neighbourhood.

3.3. Neighbour-level message interaction encoding for improving graph convolutional networks

The framework of message passing is at the core of current graph convolutional networks and is currently the dominant framework for representation learning over graphs. The essential idea of the message passing framework is to update the embedding for a node based on the message aggregated from the node's local neighbours. Therefore, this message passing update of Eq. (2) can also be described as follows.

$$\begin{aligned}
\mathbf{m}^{(k)}_{v\to u} &= Message^{(k)}(\mathbf{h}^{(k-1)}_v, \mathbf{h}^{(k-1)}_u) \\
\mathbf{m}^{(k)}_{\mathcal{N}(u)} &= \sum_{v\in\mathcal{N}(u)} \mathbf{m}^{(k)}_{v\to u} \\
\mathbf{h}^{(k)}_u &= Update^{(k)}(\mathbf{h}^{(k-1)}_u, \mathbf{m}^{(k)}_{\mathcal{N}(u)})
\end{aligned} \tag{3}$$

For each node $v \in \mathcal{N}(u)$, a message $\mathbf{m}^{(k)}_{v\to u}$ that passes from node v to node u is generated with a $Message$ function. Then an aggregated message $\mathbf{m}^{(k)}_{\mathcal{N}(u)}$ is generated by aggregating the messages from node u's local neighbours. After that, the embedding for node u is updated according to the aggregated message $\mathbf{m}^{(k)}_{\mathcal{N}(u)}$ and its previous embedding $\mathbf{h}^{(k-1)}_u$. For instance, in the GAT model, the message from a neighbour node is represented as the multiplication of the feature from this neighbour and an attention weight computed by the self-attention function, and the messages from the node's neighbours are added up together to update its embedding.

While this message passing update approach is effective, the neighbour-level message interaction information is not encoded in generated node embeddings. This information loss can accumulate as more message passing layers are added to the model, resulting in reduced performance in graph representation learning. To address this problem, we propose a neighbour-level message interaction encoding method for improving graph convolutional networks. The idea of our method is as follows: For each node $v \in \mathcal{N}(u)$, we learn an embedding between the message from node v and the aggregated message from the remaining nodes in the local neighbourhood. Then we aggregate these learned encodings and combine the aggregated encoding and the aggregated message to update the embedding for node u. In this way, the neighbour-level message interaction information is encoded in the generated node embeddings, and, therefore, the representational ability of the generated embeddings is improved.

Specifically, at each layer of message passing, we first follow Eq (3) to generate a message $\mathbf{m}_{v\to u}^{(k)}$ for each $v \in \mathcal{N}(u)$ and take the sum of the messages from u's neighbours to generate an aggregated message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$. Then for each message $\mathbf{m}_{v\to u}^{(k)}$, we learn an encoding between this message and the aggregated message from $\{\mathcal{N}(u)-v\}$, i.e., $\sum_{a\in\{\mathcal{N}(u)-v\}} \mathbf{m}_{a\to u}^{(k)}$ using a fully connected layer fc as follows:

$$\mathbf{enc}_{v,\mathcal{N}(u)-v} = fc^{(k)}\left(Concat\left(\mathbf{m}_{v\to u}^{(k)}, \sum_{a\in\{\mathcal{N}(u)-v\}} \mathbf{m}_{a\to u}^{(k)}\right)\right), \tag{4}$$

where $Concat$ denotes the concatenation operation. Because $\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \sum_{v\in\mathcal{N}(u)} \mathbf{m}_{v\to u}^{(k)} = \mathbf{m}_{v\to u}^{(k)} + \sum_{a\in\{\mathcal{N}(u)-v\}} \mathbf{m}_{a\to u}^{(k)}$, the above equation can be reformulated as follows:

$$\mathbf{enc}_{v,\mathcal{N}(u)-v} = fc^{(k)}\left(Concat\left(\mathbf{m}_{v\to u}^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} - \mathbf{m}_{v\to u}^{(k)}\right)\right). \tag{5}$$

Finally we aggregate these learned encodings and combine the aggregated encoding with the aggregated message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ to generate the updated embedding $\mathbf{h}_u^{(k)}$ as:

$$\begin{aligned}\mathbf{h}_u^{(k)} &= Update^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \sum_{v\in\mathcal{N}(u)} \mathbf{enc}_{v,\mathcal{N}(u)-v}\right)\\ &= Update^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \mathbf{enc}_{\mathcal{N}(u)}\right),\end{aligned} \tag{6}$$

where $\mathbf{enc}_{\mathcal{N}(u)} = \sum_{v\in\mathcal{N}(u)} \mathbf{enc}_{v,\mathcal{N}(u)-v}$ is the aggregated encoding. Algorithm 1 describes the node embedding generation, i.e., forward propagation, algorithm with neighbour-level message interaction information encoding. Because each of the learned encodings $\mathbf{enc}_{v,\mathcal{N}(u)-v}$ encodes the information between the message from node v and the messages from the rest of the neighbour nodes in the local neighbourhood, the neighbour-level message interaction information is integrated into the updated embedding $\mathbf{h}_u^{(k)}$.

The proposed neighbour-level message interaction encoding method is a general method that can be integrated into message passing graph convolutional networks. In this work, we validate the proposed method on two base models: the basic GCN model [14] and the GatedGCN model [15]. In the following, we describe applying the proposed neighbour-level message interaction encoding method to the two base models.

The basic GCN. The basic GCN model updates the embedding for a node by computing the average of the node's neighbouring features as follows:

$$\begin{aligned}\mathbf{m}_{\mathcal{N}(u)}^{(k)} &= \sum_{v\in\mathcal{N}(u)} \mathbf{m}_{v\to u}^{(k)}\\ &= \sum_{v\in\mathcal{N}(u)} \frac{1}{|\mathcal{N}(u)|} W^{(k)}\mathbf{h}_v\end{aligned} \tag{7}$$

where $W^{(k)}$ is a trainable weight matrix and $\mathbf{m}_{v\to u}^{(k)} = \frac{1}{|\mathcal{N}(u)|} W^{(k)}\mathbf{h}_v^{(k)}$ is the message that passes from node v to node u. With the neighbour-level message interaction encoding method, during each message-passing iteration, we update the embedding for each $u \in V$ as follows:

$$\begin{aligned}\mathbf{h}_u^{(k)} &= Update^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \sum_{v\in\mathcal{N}(u)} \mathbf{enc}_{v,\mathcal{N}(u)-v}\right)\\ &= ReLU\left(\sum_{v\in\mathcal{N}(u)} \frac{1}{|\mathcal{N}(u)|} W^{(k)}\mathbf{h}_v^{(k)} + \sum_{v\in\mathcal{N}(u)} fc\left(Concat\right.\right.\\ &\quad \left.\left.\left(\frac{1}{|\mathcal{N}(u)|} W^{(k)}\mathbf{h}_v^{(k)}, \left(\mathbf{m}_{\mathcal{N}(u)}^{(k)} - \frac{1}{|\mathcal{N}(u)|} W^{(k)}, \mathbf{h}_v^{(k)}\right)\right)\right)\right),\end{aligned} \tag{8}$$

where $ReLU$ represents the rectified linear activation function.

GatedGCN. A GatedGCN layer takes a set of node features $\{\mathbf{h}_u^{(k-1)}, u \in V\}$ and a set of edge features $\{\mathbf{e}_{uv}^{(k-1)}, (u,v) \in E\}$ as input. It employs the edge gating mechanism [28] to aggregate features passing from a node's local neighbours. The message that passes from node $v \in \mathcal{N}(u)$ to node u is defined as follows:

$$\begin{aligned}\mathbf{m}_{\mathcal{N}(u)}^{(k)} &= \sum_{v\in\mathcal{N}(u)} \mathbf{m}_{v\to u}^{(k)}\\ &= \sum_{v\in\mathcal{N}(u)} \alpha_{uv}^{(k)} \odot (F^{(k)}\mathbf{h}_v^{(k)}),\end{aligned} \tag{9}$$

where F is a trainable weight matrix, $\odot$ denotes the Hadamard product operator and $\mathbf{m}_{v\to u}^{(k)} = \alpha_{uv}^{(k)} \odot (F^{(k)}\mathbf{h}_u^{(k)})$ is the message that passes from node v to node u, and $\alpha_{uv}^{(k)}$ is the gate vector and is defined as follows:

$$\begin{aligned}\alpha_{vu}^{(k)} &= \frac{Sigmoid(\mathbf{e'}_{vu}^{(k)})}{\sum_{v\in\mathcal{N}(u)} Sigmoid(\mathbf{e'}_{vu}^{(k)}) + \epsilon},\\ \mathbf{e'}_{vu}^{(l)} &= A\mathbf{h}_u^{(k-1)} + B\mathbf{h}_u^{(k-1)} + C\mathbf{e}_{uv}^{(k-1)}\end{aligned} \tag{10}$$

where $A^{(k)}, B^{(k)}, C^{(k)}$ are trainable weight matrices and ϵ is a small constant number. With our neighbour-level message interaction encoding method, the embedding $\boldsymbol{h}_u^{(k)}$ is generated as follows:

$$\begin{aligned}\mathbf{h}_u^{(k)} &= Update^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \mathbf{enc}_{\mathcal{N}(u)}^{(k)}\right)\\ &= ReLU\left(BN\left(A^{(k)}\mathbf{h}_u^{(k)} + \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \mathbf{enc}_{\mathcal{N}(u)}^{(k)}\right)\right) + \mathbf{h}_u^{(k)},\end{aligned} \tag{11}$$

Algorithm 1 The embedding generation algorithm with the proposed neighbour-level message interaction information encoding method.

Input: Graph $G = (V, E)$; number of message passing iterations K; input node features $\{\mathbf{x}_v, \forall u \in V\}$
Output: Node embeddings $\mathbf{h}_u^{(K)}$ for all $u \in V$

$\mathbf{h}_u^{(0)} \leftarrow \mathbf{x}_u, \forall u \in V$
for $k = 1, ..., K$ **do**
  for $u \in V$ **do**
    /* aggregate messages from u's local neighbours. */
    $\mathbf{m}_{\mathcal{N}(u)}^{(k)} = Agg^{(k)}(\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\})$
      $= \sum_{v\in\mathcal{N}(u)} \mathbf{m}_{v\to u}^{(k)}$
    for $v \in \mathcal{N}(u)$ **do**
      /* learn an encoding between the message from neighbour node v and the aggregated message from the rest neighbours. */
      $\mathbf{enc}_{v,\mathcal{N}(u)-v} = fc(Concat($
        $\mathbf{m}_{v\to u}^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} - \mathbf{m}_{v\to u}^{(k)}))$
    end for
    /* combine the aggregated message and the aggregated encoding to update the embedding for u. */
    $\mathbf{h}_u^{(k)} = Update^{(k)}(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \mathbf{enc}_{\mathcal{N}(u)})$
      $= Update^{(k)}(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} + \sum_{v\in\mathcal{N}(u)} fc($
        $Concat(\mathbf{m}_{v\to u}^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} - \mathbf{m}_{v\to u}^{(k)}))$
  end for
end for

where BN denotes batch normalization, and $\mathbf{enc}_{\mathcal{N}(u)}^{(k)}$ is defined as follows:

$$\begin{aligned}\mathbf{enc}_{\mathcal{N}(u)} &= fc\left(Concat\left(\mathbf{m}_{v\to u}^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} - \mathbf{m}_{v\to u}^{(k)}\right)\right) \\ &= \sum_{v\in\mathcal{N}(u)} fc\left(Concat\left(\alpha_{uv}^{(k)} \odot \left(F^{(k)}\mathbf{h}_u^{(k)}\right), \quad \mathbf{m}_{\mathcal{N}(u)} - \alpha_{uv}^{(k)} \odot \left(\mathbf{F}^{(k)}\mathbf{h}_u^{(k)}\right)\right)\right)\end{aligned} \tag{12}$$

The edge feature $\mathbf{e}_{uv}^{(k)}$ is updated as:

$$\mathbf{e}_{uv}^{(k+1)} = ReLU\left(\mathbf{e}'^{(k)}_{ji}\right) + \mathbf{e}_{ji}^{(k)} \tag{13}$$

4. Experiments

4.1. Datasets and setup

Datasets. The proposed neighbour-level message encoding method is validated on the following six benchmark datasets [16].

- **MNIST** and **CIFAR-10**. The two datasets are superpixel graph datasets. The superpixel graphs are extracted from the images of the MNIST dataset [32] and CIFAR10 dataset [33] using the SLIC method [34]. The superpixels represent small regions of homogeneous intensity in images.
- **PATTERN** and **CLUSTER**. The two datasets contain 14 K and 12 K graphs respectively. The PATTERN dataset is used for validating the model's performance for recognizing subgraphs, and the CLUSTER dataset is used for validating the model's performance for recognizing community graphs.
- **TSP**. This dataset is constructed to evaluate graph neural networks on solving the traveling salesman problem (TSP), a classical NP-Hard combinatorial problem. We validate if or not the predicted edges of our model belong to the optimal TSP solution obtained using the Concorde solver [35].
- **ZINC**. This dataset contains 12 K samples from the ZINC molecular graph (250 K) dataset [36]. We evaluate our model for regressing the molecular property (or called constrained solubility) on this dataset.

Evaluation metrics. Following the work of Dwivedi et al. [16], the following metrics are utilized for performance evaluation according to different tasks.

- **Accuracy**. For the superpixel graph classification task on MNIST and CIFAR10, classification accuracy is used for evaluating the model performance. For the node classification task on PATTERN and CLUSTER, weighted accuracy is used for evaluating the model performance.
- **F1 score**. For the edge prediction task on TSP, due to high class imbalance, i.e., only the edges in the TSP tour are labeled as positive, the model performance is reported using the F1 score for the positive class.
- **MAE**. The MAE (mean absolute error) is used for evaluating the model's performance for regressing the molecular property on the ZINC dataset.

Baseline models. To demonstrate the superior performance of our graph network model, we compare our results against a diverse set of baseline models including MLP (multilayer perceptron), GCN [14], GraphSAGE [11], MoNet [19], GAT [12], GatedGCN [15], GIN [29], RingGNN [30] and 3WLGNN [31].

Implementation details. We closely follow the implementation details from the work of Dwivedi [16]. We adopt Adam algorithm [37] for optimizing our graph network model. We set the initial learning rate to 0.0001 and halve the learning rate when the validation loss does not decrease for 10, 20 or 30 epochs. When the learning rate decreases to a value less than 10^{-6}, we terminate the training procedure. We also apply the SSFG regularization method [24] in our model. Our model is implemented in PyTorch [38] with the DGL library [39]. The experiments are carried out on a GPU with 24GB memory GPU. We validate our model using various graph convolutional layers, e.g., 4, 8, 12, and 16. For each experiment, we validate our model 4 times using different random seeds and report the mean and standard deviation over the 4 runs.

4.2. Experimental results

Graph classification. For the superpixel graph classification task on MNIST and CIFAR10, we conduct experiments using three different graph convolutional layers, i.e., $K \in \{4, 8, 12\}$. The quantitative results are reported in Table 1, comparing our graph network model against the baseline models. We see that our model which uses GatedGCN as the base network and with 4 graph convolutional layers obtains 98.228 % and 75.001 % accuracy on MNIST and CIFAR10, respectively. Further increasing the number of graph convolutional layers to 8 and 12 results in additional performance gains. The vanilla GatedGCN achieves improved performance on the two datasets compared to the other baseline models. When compared to the vanilla GatedGCN, our 12 graph convolutional layer model improves the model performance by 1.295 % and 9.203 % on the two datasets respectively. As far as we know, our graph network model achieves better performance than previous models on the two datasets.

Node classification. For experiments on the PATTERN dataset and CLUSTER dataset, our graph network model is validated using four different graph convolutional layers, i.e., $K = 4, 8, 12, 16$. Table 2 compares the results of our model against the baseline models. For the GatedGCN as the base network, our 16 graph convolutional layer model achieves 86.643 % and 76.163 % weighted accuracy on PATTERN and CLUSTER, respectively. It can be seen that our model outperforms all the baseline models. RingGNN [30] achieves the best performance among the baseline models on PATTERN. When compared to RingGNN, our model achieves 0.398 % improved performance on PATTERN. The vanilla GatedGCN [15] obtains improved performance compared to the other baseline models on the CLUSTER dataset. Compared with the vanilla GatedGCN, applying our encoding methods yields a 2.323 % performance improvement.

Link prediction. The experimental results on link prediction on the TSP dataset are reported in Table 3. We see that our graph model which uses the GateGCN as the base network outperforms all the baseline models by a large margin, demonstrating the effectiveness of the proposed neighbour-level message interaction encoding method for improving graph convolutional network models for link prediction. The vanilla GatedGCN achieves an F1 score of 0.791, which is the best result among the baseline models. Our model with $K = 16$ outperforms the vanilla GatedGCN by 0.053. Once again, applying our method advances the state of the art performance.

Graph regression. The results on the ZINC dataset for the graph regression task are reported in Table 4. Note that during the training of our model, only node features (i.e., types of heavy atoms) are utilized. Compared with the vanilla GatedGCN-E [15], which uses both node features and edge features (i.e., bond types), our 12 layer graph network model which uses GatedGCN as the base network results in a 0.149 reduced MAE. The MoNet model [19] yields the best result among the baseline models. Compared with MoNet [19], our 12 layer graph network model which uses GatedGCN as the base network reduces the MAE from 0.292 to 0.226. The results show that integrating neighbour-level message interaction information also improves the representation learning performance for graph regression.

Fig. 2 shows the training and test accuracies/MAEs/F1 scores of our model which uses GatedGCN as the base network with different graph convolutional layers on the benchmark datasets. It can be seen that the training and test performances improve consistently as the number of graph convolutional layers increases.

Ablation study. In a layer of our model which uses GatedGCN as the base network, three features, i.e., $A\mathbf{h}_u$, $\mathbf{m}_{\mathcal{N}(u)}$ and $\mathbf{enc}_{\mathcal{N}(u)}$ are used to update node embeddings (see Eq. (11)). We carry out an ablation

Table 1
Results of our graph convolutional network on MNIST and CIFAR10 for the graph superpixel classification task. The best results of our model and the baseline models are highlighted. Refs. [11,12,14,15,19,29–31].

Model	MNIST			Model	CIFAR10		
	K	Parameters	Accuracy (↑)		K	Parameters	Accuracy (↑)
MLP	4	104K	95.340±0.138	MLP	4	104K	56.340±0.181
GraphSAGE [11]	4	104K	97.312±0.097	GraphSAGE [11]	4	105K	65.767±0.308
MoNet [19]	4	104K	90.805±0.032	MoNet [19]	4	104K	54.655±0.518
GAT [12]	4	110K	95.535±0.205	GAT [12]	4	111K	64.223±0.455
GIN [29]	4	105K	96.485±0.252	GIN [29]	4	106K	55.255±1.527
RingGNN [30]	2	105K	11.350±0.000	RingGNN [30]	2	105K	19.300±16.108
	2	505K	91.860±0.449		2	505K	39.165±17.114
	8	506K	Diverged		8	510K	Diverged
3WLGNN [31]	3	108K	95.075±0.961	3WLGNN [31]	3	109K	59.175±1.593
	3	502K	95.002±0.419		3	503K	58.043±2.512
	8	501K	Diverged		8	502K	Diverged
GCN [14]	4	101K	90.705±0.218	GCN [14]	4	102K	55.710±0.381
Ours + GCN [14]	4	357K	**91.725±0.270**	Ours + GCN [14]	4	359K	**58.455±0.002**
GatedGCN [15]	4	104K	97.340±0.143	GatedGCN [15]	4	104K	67.312±0.311
	4	144K	98.228±0.038		4	144K	75.001±0.257
Ours + GatedGCN	8	284K	98.512±0.048	**Ours + GatedGCN**	8	284K	76.395±0.101
	12	424K	**98.635±0.049**		12	424K	**76.515±0.081**

Table 2
Results of our graph convolutional network on the PATTERN dataset and CLUSTER dataset on node classification. The best result of the baseline models is highlighted in violet. Refs. [11,12,14,15,19,29–31]

Model	PATTERN			Model	CLUSTER		
	K	Parameters	Accuracy (↑)		K	Parameters	Accuracy (↑)
MLP	4	105K	50.519±0.000	MLP	4	106K	20.973±0.004
GraphSAGE [11]	4	102K	50.516±0.001	GraphSAGE [11]	4	102K	50.454±0.145
	16	503K	50.492±0.001		16	503K	63.844±0.110
MoNet [19]	4	104K	85.482±0.037	MoNet [19]	4	104K	58.064±0.131
	16	511K	85.582±0.038		16	512K	66.407±0.540
GAT [12]	4	110K	75.824±1.823	GAT [12]	4	111K	57.732±0.323
	16	527K	78.271±0.186		16	528K	70.587±0.447
GIN [29]	4	101K	85.590±0.011	GIN [29]	4	104K	58.384±0.236
	16	509K	85.387±0.136		16	518K	64.716±1.553
RingGNN [30]	2	105K	86.245±0.013	RingGNN [30]	2	105K	42.418±20.063
	2	505K	86.244±0.025		2	524K	22.340±0.000
	8	506K	Diverged		8	514K	Diverged
3WLGNN [31]	3	104K	85.661±0.353	3WLGNN [31]	3	106K	57.130±6.539
	3	503K	85.341±0.207		3	507K	55.489±7.863
	8	582K	Diverged		8	587K	Diverged
GCN [14]	4	101K	63.880±0.074	GCN [14]	4	102K	53.445±2.029
Ours + GCN [14]	4	358K	**85.615±0.049**	Ours + GCN [14]	4	359K	**58.313±0.196**
GatedGCN [15]	4	104K	84.480±0.122	GatedGCN [15]	4	104K	60.404±0.419
	16	502K	85.568±0.088		16	503K	73.840±0.326
	4	143K	85.669±0.034		4	144K	63.212±0.130
Ours + GatedGCN	8	283K	86.336±0.136	**Ours + GatedGCN**	8	284K	73.055±0.135
	12	423K	86.601±0.084		12	424K	75.458±0.108
	16	563K	**86.643±0.047**		16	564K	**76.163±0.101**

Table 3
Experimental results on the TSP dataset for link prediction. OOM denotes out of memory. Refs. [11,12,14,15,19,29–31]

Model	K	TSP Parameters	TSP F1 (↓)
MLP	4	97K	0.544±0.0001
GraphSAGE [11]	4	90K	0.665±0.003
MoNet [19]	4	99K	0.641±0.002
GAT [12]	4	96K	0.671±0.002
GIN [29]	4	99K	0.656±0.003
RingGNN [30]	2	107K	0.643±0.024
	2	508K	0.704±0.003
	8	506K	Diverged
3WLGNN [31]	3	106K	0.694±0.073
	3	507K	0.288±0.311
	8	509K	OOM
GCN [14]	4	96K	0.630±0.0001
Ours + GCN	4	269K	**0.651±0.0002**
GatedGCN [15]	4	98K	0.791±0.003
Ours + GatedGCN	4	132K	0.811±0.001
	8	253K	0.832±0.001
	12	373K	0.841±0.001
	16	495K	**0.844±0.001**

Table 4
Experimental results on the ZINC dataset for the graph regression task. Note that only node features are used for training our model. Refs. [11,12,14,15,19,29–31]

Model	K	ZINC Parameters	ZINC MAE (↓)
MLP	4	109K	0.706±0.006
GraphSAGE [11]	4	95K	0.468±0.003
	16	505K	0.398±0.002
MoNet [19]	4	106K	0.397±0.010
	16	504K	0.292±0.006
GAT [12]	4	102K	0.475±0.007
	16	531K	0.384±0.007
GIN [29]	4	103K	0.387±0.015
	16	510K	0.526±0.051
RingGNN [30]	2	98K	0.512±0.023
3WLGNN [31]	3	102K	0.407±0.028
GCN [14]	4	103K	0.459±0.006
Ours + GCN	4	356K	**0.375±0.002**
GatedGCN [15]	4	106K	0.435±0.011
GatedGCN-E [15]	4	106K	0.375±0.003
Ours + GatedGCN	4	146K	0.268±0.005
	8	285K	0.233±0.004
	12	425K	**0.226±0.002**

study to show the significance of each feature on the model's overall performance. Note that when only $A\mathbf{h}_u$ and $\mathbf{m}_{\mathcal{N}(u)}$ are used, our graph network model is equivalent to the vanilla GatedGCN model. As shown in Table 5, we see that integrating the neighbour-level message interaction information consistently improves the graph representation learning performance on all the benchmark datasets. It can also be seen that the use of $\mathbf{enc}_{\mathcal{N}(u)}$ performs better than the use of $\mathbf{m}_{\mathcal{N}(u)}$. The results demonstrate that encoding neighbour-level message interaction information is a generic method that consistently improves the representation learning performance on graphs.

Discussion. The context-aware subtraction $\mathbf{m}_{\mathcal{N}(u)} - \mathbf{m}_{v \to u}$ emphasizes relative differences between neighbour nodes. The term $\mathbf{m}_{\mathcal{N}(u)} - \mathbf{m}v \to u$ acts as a high-pass component, preventing node embeddings from becoming similar to one another. The results in Tables 1–4 show that the model performance improves as the layer number is increased to 16. This suggests that our method helps mitigate the oversmoothing problem (low-pass filtering), implying that our encoding method preserves mid/high-frequency discriminative signals.

5. Conclusion

This paper proposes a neighbour-level message interaction information encoding method for improving graph convolutional networks. At each layer of message passing, we first follow the conventional framework to generate an aggregated message for a node by aggregating

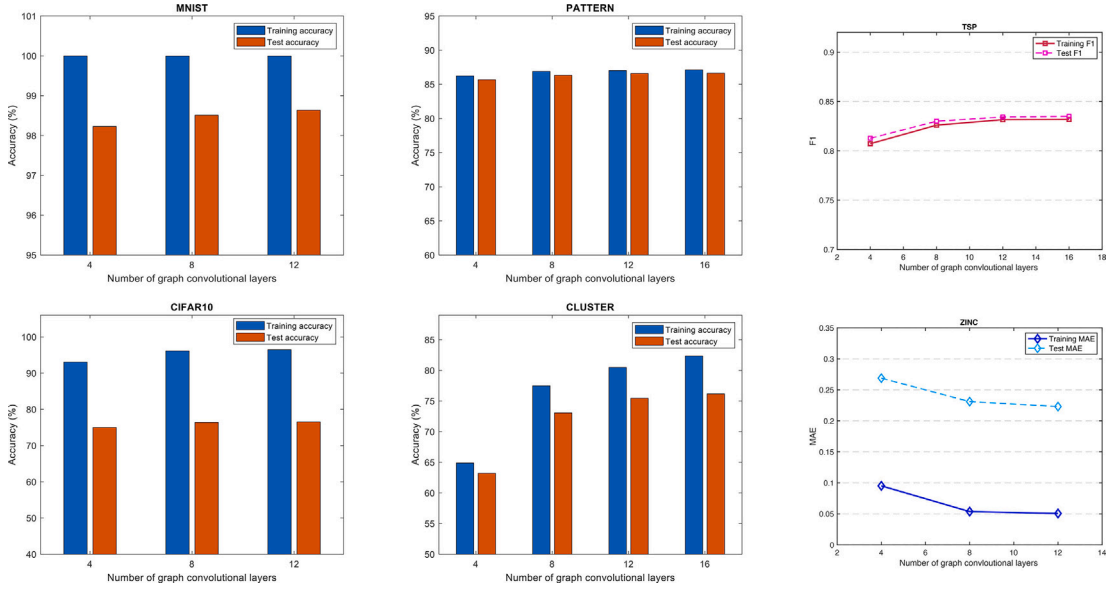

Fig. 2. Performance our model using different graph convolutional layers.

Table 5
Ablation study: importance of each feature component for generating a node's embedding (see Eq. (11)) on the model performance.

Method	Accuracy (↑)				F1 (↑)	MAE (↓)
	MNIST ($K=12$)	CIFAR10 ($K=12$)	PATTERN ($K=16$)	CLUSTER ($K=16$)	TSP ($K=16$)	ZINK ($K=12$)
$A\mathbf{h}_u + \mathbf{m}_{\mathcal{N}(u)}$	98.140	72.032	85.723	75.245	0.818	0.281
$A\mathbf{h}_u + \mathbf{enc}^{(k)}_{\mathcal{N}(u)}$	98.490	76.383	86.589	76.158	0.840	0.227
$\mathbf{m}_{\mathcal{N}(u)} + \mathbf{enc}_{\mathcal{N}(u)}$	98.573	76.497	86.620	76.056	0.843	0.226
$A\mathbf{h}_u + \mathbf{m}_{\mathcal{N}(u)} + \mathbf{enc}_{\mathcal{N}(u)}$	98.635	76.515	86.643	76.163	0.844	0.226

information from the node's local neighbourhood. For each of the node's local neighbours, we learn an encoding between the message from the neighbour node and the aggregated message from the rest of the neighbour nodes using a fully connected layer. These learned encodings are aggregated to generate a neighbour-level message interaction encoding. Finally, the sum of the aggregated message and neighbour-level message encoding is taken to generate an updated embedding for the node. In this way, neighbour-level message interaction information is integrated into the generated node embeddings, and, therefore, the representational ability of the generated embeddings is improved. We experimentally evaluated our graph network model on four graph-based tasks, including superpixel graph classification, node classification, edge prediction and graph regression, on six recently released benchmark datasets, including MNIST, CIFAR10, PATTERN, CLUSTER, TSP and ZINC. The results showed that integrating the proposed neighbour-level message interaction information encoding method is a generic approach to improve the representation learning performance on graph-structured data. We demonstrated that integrating neighbour-level message interaction information achieves new state-of-the-art performance on the benchmark datasets.

CRediT authorship contribution statement

Haimin Zhang: Writing – original draft, Validation, Software, Methodology, Conceptualization. **Min Xu:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank the editors and reviewers for reviewing this manuscript. This work was supported by the Australian Research Council under the Linkage Projects Grant LP210100129.

Data availability

The data are publicly available.

References

[1] W.L. Hamilton, Graph representation learning, Synth. Lect. Artif. Intell. Mach. Learn. 14 (3) (2020) 1–159.
[2] X. Liu, S. Meng, Q. Li, Q. Liu, Q. He, D. Ramesh, L. Qi, FDGNN: feature-aware disentangled graph neural network for recommendation, IEEE Trans. Comput. Soc. Syst. 11 (1) (2023) 1372–1383.
[3] H. Huang, H. Tian, X. Zheng, X. Zhang, D.D. Zeng, F.-Y. Wang, CGNN: a compatibility-aware graph neural network for social media bot detection, IEEE Trans. Comput. Soc. Syst. 11 (5) (2024) 6528–6543.
[4] J. Li, L. Wu, Y. Du, R. Hong, W. Li, Dual graph neural networks for dynamic users' behavior prediction on social networking services, IEEE Trans. Comput. Soc. Syst. 11 (5) (2024) 7020–7031.
[5] C. Shi, Y. Mao, Y. Shen, W. Xiong, F. Liu, C. Li, C. Wang, MTT-DynGL: towards multi-dimensional topology-oriented time-series dynamic graphs learning model, in: 2023 IEEE International Conference on Data Mining (ICDM), IEEE, 2023, pp. 1313–1318.
[6] Q. Xu, F. LiU, Z. Fu, A. Zhou, J. Qi, AeS-GCN: attention-enhanced semantic-guided graph convolutional networks for skeleton-based action recognition, Comput. Animat. Virtual Worlds 33 (3–4) (2022) e2070.
[7] Z. Fu, F. Liu, J. Zhang, H. Wang, C. Yang, Q. Xu, J. Qi, X. Fu, A. Zhou, SAGN: semantic adaptive graph network for skeleton-based human action recognition, in: Proceedings of the 2021 international conference on multimedia retrieval, 2021, pp. 110–117.
[8] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. 2005 IEEE international joint conference on neural networks, IEEE, Vol. 2, 2005, 729–734.
[9] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[10] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: International Conference on Learning Representations, 2014, pp. 1–14.
[11] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
[12] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, Int. Conf. Learn. Represent. (2018) https://openreview.net/forum?id=rJXMpikCZ.
[13] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, P. Honeine, Analyzing the expressive power of graph neural networks in a spectral perspective, in: International Conference on Learning Representations, 2020.
[14] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR2017), 2017.
[15] X. Bresson, T. Laurent, Residual gated graph convnets, arXiv preprint arXiv:1711.07553, 2017.
[16] V.P. Dwivedi, C.K. Joshi, T. Laurent, Y. Bengio, X. Bresson, Benchmarking graph neural networks, arXiv preprint arXiv:2003.00982, 2020.
[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.
[18] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[19] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern recognition, 2017, pp. 5115–5124.
[20] S.Y. Lee, F. Bu, J. Yoo, K. Shin, Towards deep attention in graph neural networks: problems and remedies, Int. Conf. Mach. Learn. (2023) 18774–18795.
[21] H. Gao, X. Han, J. Huang, J.-X. Wang, L. Liu, PatchGT: transformer over non-trainable clusters for learning graph representations, in: Learning on Graphs Conference, PMLR, 2022, pp. 27.
[22] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 1725–1735.
[23] L. Zhao, L. Akoglu, PairNorm: tackling oversmoothing in gnns, in: International Conference on Learning Representations, 2020, pp. 1–9.
[24] H. Zhang, M. Xu, G. Zhang, K. Niwa, SSFG: stochastically scaling features and gradients for regularizing graph convolutional networks, IEEE Trans. Neural Netw. Learn. Syst. 35 (2) (2022) 2223–2234.
[25] J. Zhu, C. Gao, Z. Yin, X. Li, J. Kurths, Propagation structure-aware graph transformer for robust and interpretable fake news detection, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 4652–4663.
[26] C. Gao, J. Zhu, F. Zhang, Z. Wang, X. Li, A novel representation learning for dynamic graphs based on graph convolutional networks, IEEE Trans. Cybern. 53 (6) (2022) 3599–3612.
[27] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.
[28] D. Marcheggiani, I. Titov, Encoding sentences with graph convolutional networks for semantic role labeling, arXiv preprint arXiv:1703.04826, 2017.
[29] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? 2019.
[30] Z. Chen, S. Villar, L. Chen, J. Bruna, On the equivalence between graph isomorphism testing and function approximation with GNNs, Adv. neural inf. process. syst. 32 (2019) 1–9.
[31] H. Maron, H. Ben-Hamu, H. Serviansky, Y. Lipman, Provably powerful graph networks, arXiv preprint arXiv:1905.11136, 2019.
[32] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[33] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, Toronto (2009).

[34] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, SLIC superpixels compared to state-of-the-art superpixel methods, IEEE Trans. Pattern Anal. Mach. Intell. 34 (11) (2012) 2274–2282.
[35] D. Applegate, R. Bixby, V. Chvatal, W. Cook, Concorde tsp solver (2006).
[36] J.J. Irwin, T. Sterling, M.M. Mysinger, E.S. Bolstad, R.G. Coleman, ZINC: a free tool to discover chemistry for biology, J. Chem. Inf. Model. 52 (7) (2012) 1757–1768.
[37] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.
[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).
[39] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, et al., Deep graph library: towards efficient and scalable deep learning on graphs, arXiv preprint arXiv:1909.01315, 2019.