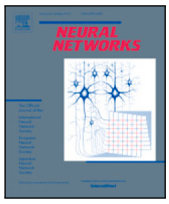
Neural Networks 190 (2025) 107658

Contents lists available at ScienceDirect

Neural Networks

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



Full Length Article

Adaptive receptive field graph neural networks

Hepeng Gao [a], Funing Yang [a,*], Yongjian Yang [a], Yuanbo Xu [a], Yijun Su [b]

[a] Jilin University, Changchun, 130012, Jilin, China
[b] JD Intelligent Cities Research, Beijing, 100007, China

ARTICLE INFO

Dataset link: https://github.com/wn13/ADRPGNN

Keywords:
Graph Neural Network
Over-smooth issue
Node classification

ABSTRACT

Graph Neural Networks (GNNs) have drawn increasing attention in recent years and achieved outstanding success in many scenarios and tasks. However, existing methods indicate that the performance of representation learning drops dramatically as GNNs deepen, which is attributed to **over-smoothing representation**. To handle the above issue, we propose an adaptive receptive field graph neural network (ADRP-GNN) that aggregates information by adaptively expanding receptive fields with a monolayer graph convolution layer, avoiding deepening to result in the over-smoothing issue. Specifically, we first present a Multi-hop Graph Convolution Network (MuGC) that captures the information of the nodes and their multi-hop neighbors with only one layer, preventing frequent passing messages between nodes from the over-smoothing issue. Then, we design a Meta Learner that realizes the adaptive receptive field for each node to select related neighbor information. Finally, a Backbone Network is employed to enhance the architecture's learning ability. In addition, our architecture adaptively generates receptive fields instead of handcrafting stacked layers, which can integrate existing GNN frameworks to fit various scenarios. Extensive experiments indicate that our architecture is effective for the over-smoothing issue and improves accuracy by 0.52% to 6.88% compared to state-of-the-art methods on node classification tasks on eight datasets.

1. Introduction

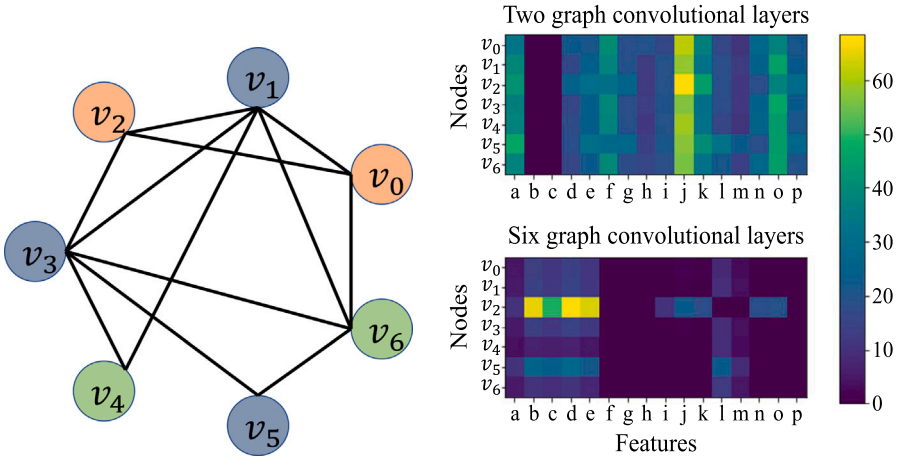
Graphs are of the essence for an irregular domain, and they are omnipresent in most areas (Duvenaud et al., 2015; Monti, Bronstein, & Bresson, 2017; Zhang et al., 2018). Graph Neural Networks (GNNs), a practical framework for learning graph representation, learn the node and edge representations and graph structures to deal with graph-based data (Jiang, Ji, & Li, 2019). The idea of GNNs is to aggregate the features within receptive fields by stacking multiple layers, thus achieving the fusion of the features and the structure. The receptive field is a region in the input space that influences the computation of a particular node's representation within the graph, encompassing all the nodes and edges that contribute to the node's final representation after multiple message-passing steps. Despite the success of graph modeling, a common issue, over-smoothing representation, is inevitable in GNNs, where the node representations become more homologous and indistinguishable as the number of stacking graph convolutional layers increases in GNN frameworks. Hence, these homologous representations fail to support downstream tasks.

To introduce the over-smoothing representation situation, we give a visualized toy example by deploying GCN (Kipf & Welling, 2017) for the node classification task as shown in Fig. 1. When GCN goes deep (layer=6), the learned node representations of multiple categories are too similar to classify, which is a typical over-smoothing issue. However, in scenarios and tasks where GNNs are used, it is necessary to stack enough graph convolution layers to enlarge the receptive fields for structural information. In summary, a deeper GNN obtains more structured information, but it has to solve or relieve the over-smoothing issue to guarantee the task's performance, making it a dilemma in existing GNNs.

To our knowledge, several works have been devoted to tackling the over-smoothing issue. Among them, most existing works (Chen, Wei, Huang, Ding, & Li, 2020; Giraldo, Skianis, Bouwmans, & Malliaros, 2023; Hu et al., 2020; Liu, Gao, & Ji, 2020; Liu, Zhou, et al., 2023; Ma, Wang, Chen, & Song, 2021) focus on how to go deep, where GNN architectures enlarge receptive fields for sufficient neighbor information and relieve the over-smoothing issue by various methods. In addition, the idea of N-GCN (Abu-El-Haija, Kapoor, Perozzi, & Lee, 2020) is how to go wide, where graph convolutional layers aggregate information from various subgraph structures to improve performance and relieve the over-smoothing issue, but it does not investigate the over-smoothing issue. However, these works still fall short due to the following challenges:

* Corresponding author.
E-mail addresses: gaohepeng13@hotmail.com (H. Gao), harptomato@163.com (F. Yang), yyj@jlu.edu.cn (Y. Yang), yuanbox@jlu.edu.cn (Y. Xu), suyijun.ucas@gmail.com (Y. Su).

https://doi.org/10.1016/j.neunet.2025.107658
Received 21 September 2024; Received in revised form 6 May 2025; Accepted 19 May 2025
Available online 5 June 2025



(a) Graph structure (b) Node embedding

Fig. 1. (a) shows an unweighted undirected subgraph. The colors of nodes indicate labels, while the edges represent adjacency relations. (b) shows node embedding visualizations with two and six graph convolutional layers.

- **Model Depth.** It is a primary concern for representation learning how to balance the model's depth and the over-smoothing issue. By stacking layers, the receptive field of a node with a high degree could lead to over-smoothing (Chen, Wei et al., 2020). In other words, nodes frequently pass messages, making the representations indistinguishable.
- **Adaptivity.** In traditional methods, nodes have the same receptive field, i.e., the same number of stacking graph convolution layers, so target nodes could not precisely obtain related nodes to learn structure information or introduce noise. In most cases, each node in a graph has its own suitable receptive field instead of all nodes having a fixed receptive field.

To bridge the gap, we propose a novel graph convolutional architecture, the adaptive receptive field graph neural network (ADRP-GNN), to generate receptive fields of nodes and aggregate the features of nodes and their multi-hop neighbors with a monolayer. More specifically, our architecture consists of three components: (1) **Multi-hop Graph Convolution Network** (MuGC) takes into account the features within K-hop neighbors. It decouples going deep and enlarging receptive fields and obtains sufficient receptive fields as a monolayer graph convolutional layer, which avoids GNNs going deep and relieves the over-smoothing issue. (2) **Meta Learner** predicts the suitable receptive fields K to achieve adaptive receptive fields for each node and avoid noise. (3) **Backbone Network**, stacking fully connected networks, enhances learning ability to support the shallow MuGC. The contributions of this paper are summarized as follows:

- We propose a novel graph convolutional architecture with a new perspective that decouples receptive fields from stacking layers and directly obtains distant node information. MuGC goes wide instead of deep to enlarge the receptive fields and relieve the over-smoothing issue.
- We propose a Meta Learner to predict the receptive fields of nodes, addressing the problem of noise and insufficient information during GNN messaging. Each node adaptively generates a suitable receptive field instead of all nodes having a fixed receptive field.
- We conduct extensive experiments to evaluate our model on eight real-world graph datasets. Our architecture improves accuracy by 0.52% to 6.88% compared to state-of-the-art methods on node classification tasks.

2. Related work

2.1. Graph tasks

GNNs have been widely applied to various tasks and scenarios across multiple domains. Unlike the tasks of CNNs, which increase the feature channels and reduce the size of the feature map, GNNs are designed to learn effective representations for nodes, edges, or entire graphs to facilitate downstream tasks. Graph learning tasks can be broadly categorized into three types: (1) Node-level tasks aim to learn node representations for downstream applications, e.g., node classification (Liu, Zhan, et al., 2023; Luan et al., 2023; Tan, Guo, Ding, & Liu, 2023; Zhao, Jin, et al., 2024), node clustering (Liu, Liang, et al., 2023; Liu, Yang, et al., 2023; Pan & Kang, 2023; Tsitsulin, Palowitch, Perozzi, & Müller, 2023), etc. These tasks are prevalent in social network analysis, recommendation systems, and molecular property prediction. (2) Edge-level tasks focus on modeling pairwise relationships, e.g., link prediction (Gregucci, Nayyeri, Hernández, & Staab, 2023; Li et al., 2023; Liu, Li, Fiumara, & De Meo, 2023; Tan, Zhang, et al., 2023; Zhang et al., 2023). These are particularly relevant in applications, including knowledge graph completion, protein-protein interaction networks, fraud detection, etc. (3) Graph-level tasks seek to generate holistic representations of entire graphs, e.g., graph classification (Guo & Mao, 2023; Ju et al., 2024; Luo, Shi, & Wu, 2025; Ma, Hu, Ge & Zhang, 2023; Yin et al., 2023). These tasks are widely used in drug discovery, material science, etc.

In addition, to capture structural dependencies, two main GNN paradigms have emerged: (1) Spectral methods (Defferrard, Bresson, & Vandergheynst, 2016; Kipf & Welling, 2017; Li, Wang, Zhu, & Huang, 2018; Lu et al., 2024; Wang & Zhang, 2022; Yang et al., 2022) define graph convolutions based on spectral graph theory, effectively modeling global patterns but often lacking generalizability across graphs. (2) Spatial methods (Donnat & Jeong, 2023; Ma, Lin, et al., 2023; Velickovic et al., 2018; Xu, Hu, Leskovec, & Jegelka, 2019; Zhang, Cheng, Yuan, & Zhang, 2024; Zhao, Zheng, et al., 2024) define convolutions over local neighborhoods, offering better scalability and adaptability to large or dynamic graphs, but rely on carefully designed message-passing schemes.

2.2. Over-smoothing issue in GNNs

The issue that representations of neighboring nodes are too close to classify as GNNs go deep is called over-smoothing and has been demonstrated by the literature (Bo, Wang, Shi, & Shen, 2021; Li, Han,

& Wu, 2018; Yang, Wang, Gu, Cao, & Niu, 2021). MADGap (Chen, Lin, et al., 2020) for measuring the over-smoothness of the graph node representations is further proposed. Many methods have emerged to solve the over-smoothing issue. GRAND (Feng et al., 2020) designs a self-supervised task that randomly masks node features to generate multiple graphs and utilizes consistency regularization to improve the model's performance. GCLN (Hu et al., 2020) characterizes and enhances indistinguishable features by fusing the corresponding contextual information from the contracting side to the deeper layers. N-GCN enlarges the feature channels of each layer by random walk generating subgraphs. CGNN (Xhonneux, Qu, & Tang, 2020) builds the connection between recent GNNs and traditional dynamic systems, which is robust to over-smoothing. CurvDrop (Liu, Zhou, et al., 2023) integrates the Discrete Ricci Curvature into graph neural networks to enable more expressive graph models. ACMP (Wang, Yi, Liu, Wang, & Jin, 2023) which has a simple implementation with a neural ODE solver can propel the network depth up to one hundred of layers with a theoretically proven strictly positive lower bound of the Dirichlet energy. Our architecture proposes a single graph convolutional layer to relieve deep GNNs leading to over-smoothing.

2.3. Adaptive receptive fields

To solve the over-smoothing issue and obtain better performance, adaptive approaches are widely applied to GNN. ADSF (Zhang, Zhu, Wang, & Zhang, 2020) proposes an adaptive structural fingerprint model to encode complex topological and structural information of the graph to improve graph representation learning. STAR-GNN (Ma et al., 2021) proposes a method based on anonymous random walks to model the local structural embedding of nodes, and construct receptive fields discretely and adaptively based on the mutual information of the neighboring nodes and target nodes. MpnnDRF (Wang, Di, & Chen, 2023) to improve the ability of the message passing neural network by capturing the receptive field in different graphs. Our architecture proposes that Meta Learner select suitable receptive fields for nodes to adapt to different scenarios and tasks.

3. Preliminaries

In this section, we give the mathematical definitions and problem statements discussed in this paper for convenience.

A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ is a set of N nodes and $\mathcal{E} = \{e_{ij} | 1 \le i, j \le N\}$ is a set of edges. The attribute set $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ has a one-to-one correspondence with the node set $\mathcal{V}$, where x_i represents the feature vector of node v_i. The downstream task labels are defined as $\mathcal{Y} = \{y_1, y_2, \dots, y_N\}$.

Problem Definition. Given $\mathcal{G}$, the goal is to learn distinguishable representation vectors $\mathcal{Z} = \{z_1, z_2, \dots, z_N\}$ of nodes $\mathcal{V}$ and using $\mathcal{Z}$ to handle downstream node classification tasks, i.e., $f : \mathcal{Z} \to \hat{\mathcal{Y}}$ where $\hat{\mathcal{Y}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N\}$ is predicted downstream task labels. Our framework focuses on relieving over-smoothing representations of nodes to learn distinguishable and effective representations when nodes expand receptive fields.

4. Methodology

In this section, we describe the proposed overall architecture as shown in Fig. 2. Our proposed ADRP-GNN takes the graph structure and node features as input and outputs the node representations for addressing downstream tasks. It consists of three main steps: (a) **Multi-hop Graph Convolution Network.** The main function of this component is to aggregate the information within K-hop neighbors with a monolayer graph convolution. (b) **Meta Learner.** This component predicts the receptive field K and provides it to MuGC. (c) **Backbone Network.** This component increases the depth of the architecture to enhance learning ability.

In addition, we propose a meta-loss function and a task-loss function to optimize ADRP-GNN. More details of our architecture and optimizations are illustrated in the following. Further, we analyze the time consumption of our architecture and reduce computational consumption.

4.1. Backbone network

Compared with the traditional method, using stacking layers to enlarge the range of aggregated neighbors increases the number of nonlinear layers of the models, which also enhances the ability of feature extraction and fusion. However, MuGC using only one layer aggregates the features of multi-hop neighbors, whose ability is limited to extract features and fuse information.

To this end, we present the Backbone Network, consisting of stacking fully connected layers, to overcome the drawback. In addition, Backbone Network compresses the node representation dimension to decrease the number of architecture parameters and accelerate the computational speed. It is expressed as:

$$h_i^{(t+1)} = \sigma(f^t_{\text{backbone}}(h_i^t)), \tag{1}$$

where, f^t_{backbone} is layer t ($t \in \{0, 1, 2, \dots\}$) of Backbone Network, and h_i^t and $h_i^{(t+1)}$ are the input and output of layer t, respectively. h_i^0 is the feature x_i of node v_i and the output $\mathbf{H}$ of Backbone Network inputs the MuGC.

4.2. Multi-hop Graph Convolution Network

We propose MuGC, a graph convolution layer that aggregates features of neighbors to update the representation of target nodes based on graph structure. Compared with traditional GNNs, our proposed MuGC directly aggregates distant neighbors without messages passing repeatedly.

In traditional GNNs, target nodes aggregate distant neighbors' information by immediate neighbors passing. Therefore, the node representations contain the information that they require and pass, which leads to the representations being indistinguishable. As shown in Fig. 3(a), the target node v_0 repeatedly aggregates representations of directly connected nodes v_1, v_2, and v_6 to obtain graph structure information. The representations of nodes v_1, v_2, and v_6 could consist of information they and node v_0 need, i.e., $\bar{v_0}$, which results in nodes v_0, v_1, v_2, and v_6 being similar. Meanwhile, node v_0 passes messages to other nodes, when the other nodes as target nodes, e.g., v_1, v_2, and v_6. With receptive fields enlarging, the representations contain more information that the node does not need but is required by neighbors within receptive fields and become over-smoothing.

In summary, existing methods are serial aggregation, where nodes contain information that their neighbors within the same receptive field need, which results in over-smoothing representations. Existing methods to mitigate the over-smoothing problem while capturing multi-hop information include the following approaches: (1) Edge manipulation (DropEdge (Rong, Huang, Xu, & Huang, 2019), GeniePath (Liu et al., 2019) and SJLR (Giraldo et al., 2023)), where edges are randomly added or removed to prevent excessive message passing between nodes. (2) Residual connections (GCNII (Chen, Wei et al., 2020)) and aggregation of intermediate layer outputs (DAGNN (Liu et al., 2020)), which combine both shallow and deep features to generate node representations, thereby enhancing multi-hop information. (3) Topological enhancement (SMEGCN (Jiang, Yang, Wen, Su, & Huang, 2022)), where the topology used during the aggregation process is enhanced to capture richer multi-hop information and address the over-smoothing issue.

To address the above issue, we design the MuGC, a message-passing method that directly obtains information from distant neighbors instead of passing it repeatedly. The MuGC aggregates the neighbors' information of each hop separately and combines the aggregated information

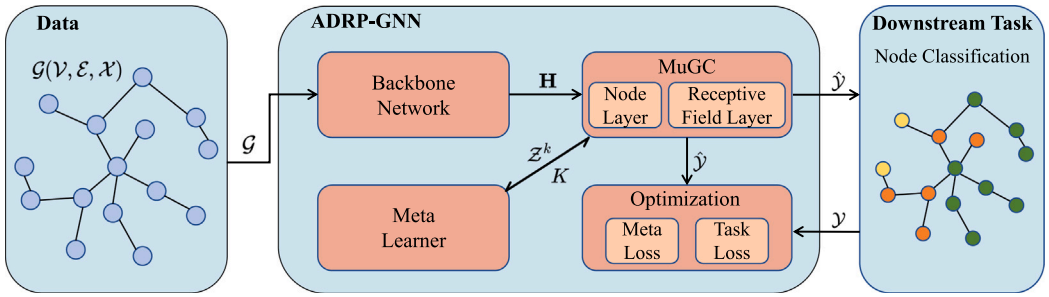


Fig. 2. An overall architecture of ADRP-GNN. The input graph is initially processed by a backbone network to extract node features **H**. Subsequently, the Node Layer separately extracts information of multi-hop neighbors, providing $\mathcal{Z}^k$ as input to the Meta Learner for determining the receptive fields K. The Receptive Field Layer then aggregates $\mathcal{Z}^k$ based on K to construct the node representations $\mathcal{Z}$ and infer the downstream task labels $\hat{\mathcal{Y}}$. The optimization module comprises Meta Loss and Task Loss, responsible for optimizing the Meta Learner and the others, respectively.

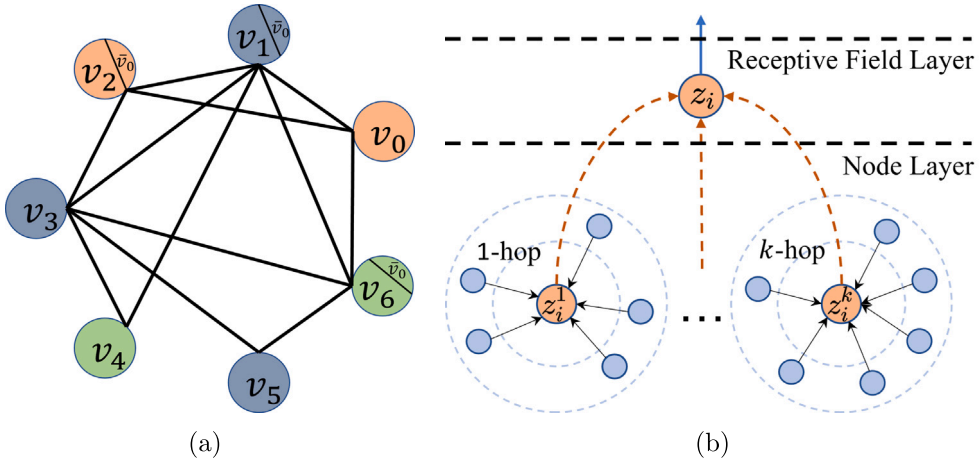


Fig. 3. (a) The message passing of traditional GNNs where node v_0 is the target node and nodes v_1, v_2, and v_6 are the 1-hop neighbor nodes. The $\bar{v}_0$ denotes information that v_0 needs in the distance in the message passing. (b) Explanation of MuGC aggregating process. The blue circles are neighbors while the orange nodes are target nodes i.

of all hops to update target nodes. Specifically, the MuGC consists of a Node Layer and a Receptive Field Layer, where the former separately extracts information of each hop, while the latter combines information of hops within receptive fields, as shown in Fig. 3(b).

4.2.1. Node layer

Inspired by the traditional GNNs enlarging the receptive field, the Node Layer extracts information of neighbors within K-hop, as shown in Fig. 3(b). The mathematical expression of Node Layer is defined as

$$z_i^k = \begin{cases} W_n h_i, & k = 0 \\ F_G(W_n h_i, \{W_n h_j, v_j \in \mathcal{N}_i^k\}), & k > 0 \end{cases} \tag{2}$$

where i and k indicate the node index and the hop index, respectively. W_n is a weight matrix to transform and compress the initial features, and $\mathcal{N}_i^k$ denotes the set of nodes whose minimum distance is k away from the target node v_i.

F_G is a network aggregating single-hop neighbors' information, which could be designed based on tasks or adopt existing message-passing methods, e.g., GAT (Velickovic et al., 2018), GCN (Kipf & Welling, 2017), etc. Target nodes are regarded as their 0-hop neighbors, i.e., $v_i \in \mathcal{N}_i^0$. We utilize the attention-based method as the message-passing method.

Through the above process, we obtain a set of representations $\mathcal{Z}_i^k = \{z_i^0, z_i^1, \ldots, z_i^{k_i}\}$ within the receptive field k_i of the target node v_i.

4.2.2. Receptive field layer

The Node Layer generates the $0, 1, \ldots, k$-hop representations whose information does not overlap. Furthermore, the vital issue, in accomplishing the aggregation, is exploring how much each hop representation affects each other and contributes to the final representations. The Receptive Field Layer, as shown in Fig. 3(b), combines the information within the receptive field. The target node v_i representation is calculated as:

$$z_i = F_u(\{z_i^0, z_i^1, \ldots, z_i^{k_i}\}), \tag{3}$$

where F_u is an attention-based function aggregating information of hops, which can handle variable-length sequences adaptively and in accord with the message-passing mechanism. The attention-base function learns the importance of each hop's information for the target nodes. The final prediction $\hat{y}_i$ for the downstream task is made using z_i, or alternatively, a linear layer is added to map it.

4.3. Meta learner

Considering existing works utilize stacking graph convolutional layers to enlarge receptive fields. However, they cannot select the suitable receptive field for each node, rather the receptive fields are fixed for the reason that the model architectures are static. Obviously, nodes have different influences and contributions to other nodes on representation learning. With too small a receptive field, the target node cannot obtain adequate structural information and, conversely, aggregating irrelevant nodes results in the introduction of noise. In addition, previous works manually design the depth of the model for various scenarios and tasks based on experiences.

Meta Learner presents a new perspective on generating adaptive receptive fields K that are optimal for each node. Meta Learner extracts meta knowledge from node features and graph structures and predicts receptive fields K to MuGC, which indicates f_{meta} and consists of two parts: f_{rnn} and f_{score}.

Inspired by the order of traditional GNNs in message passing, we utilize the LSTM to learn how neighbors within k-hop affect target nodes. From the perspective of the graph structure, distant neighbors affect and connect to target nodes via immediate neighbors. Hence, we apply the gate mechanism to join and remove the distant information based on the immediate information. The f_{rnn} obtains information of larger receptive fields, which is calculated as:

$$t_i^k = f_{\text{rnn}}(z_i^k). \tag{4}$$

Finally, we score the receptive fields and select the optimal, which is formulated as follows:

$$k_i = \arg\max_{k_i^*}(f_{\text{score}}(t_i^{k^*})), \tag{5}$$

where f_{score} is a fully connected network to predict the score of the receptive field $k^*(k^* = 0, 1, 2, \ldots)$. After that, we select the optimal receptive field for the target node v_i at the maximum score as k_i, which means that features of neighbors within k_i-hops are aggregated when the target node v_i generates the representation.

4.4. Optimization

Our architecture provides an end-to-end architecture to learn deep representations and address downstream tasks. Specifically, we design two loss functions to optimize the parameters of the architecture, i.e., the task-loss function and the meta-loss function, where the former aims to learn representations and predict labels, while the main objective of the latter is to select the optimal receptive fields.

4.4.1. Task-loss function

Supposing the task-loss function for architecture training is $\mathcal{L}_{\text{Task}}$, which measures the performance of a model on a downstream task. The task-loss function optimizes the MuGC and the Backbone network by minimizing the difference between the ground truth and predicted values. The task-loss function takes architecture outputs and downstream task labels as input, calculated by:

$$\mathcal{L}_{\text{Task}} = \frac{1}{n^t}\sum_{i=0}^{n^t} l_t(y_i, \hat{y}_i|k_i), \tag{6}$$

where n^t is the number of instances, y_i is the ground truth, $\hat{y}_i$ is the prediction value, and k_i is the receptive fields that Meta Learner predicts. l_t varies according to the downstream task. We take cross-entropy as l_t for the node classification.

4.4.2. Meta-loss function

The strategy, which selects receptive fields, is a discrete search space and cannot be optimized directly with the task-loss function. For the meta-loss function $\mathcal{L}_{\text{Meta}}$, we introduce a probabilistic score to evaluate the task prediction accuracy under the receptive field k_i. To select the optimal receptive field, we design l_m as cross-entropy to fit curves with hop representations and receptive fields as independent variables and downstream task accuracy as dependent variables. We optimize the parameters of the Meta Learner by minimizing $\mathcal{L}_{\text{Meta}}$, which is expressed as:

$$\mathcal{L}_{\text{Meta}} = \frac{1}{n^m}\sum_{i=0}^{n^m} l_m(\mathbb{1}(y_i, \hat{y}_i|k_i), f_{\text{meta}}(z_i|k_i)), \tag{7}$$

where n^m is the number of instances. $\mathbb{1} \in \{0, 1\}$ is an indicator function that evaluates to 1 if and only if node v_i is the correct classification.

4.4.3. Training

Algorithm 1 Training algorithm of ADRP-GNN.

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$; Task Label $\mathcal{Y}$.
1: initialize all trainable parameters
2: **while** stopping criteria is not met **do**
3: Meta Learner calculates the receptive field K
4: forward-backward on $\mathcal{L}_{\text{Task}}$ by $\mathcal{G}$, $\mathcal{Y}$, and K
5: update ω_{mu} and ω_{bk}
6: randomly select the receptive field K
7: calculate $\mathcal{L}_{\text{Task}}$
8: forward-backward on $\mathcal{L}_{\text{Meta}}$ by $\mathcal{L}_{\text{Task}}$, K
9: update ω_{mt}
10: **end while**
11: **return** output parameters of learned ADRP-GNN

The parameters of architecture are trained by $\mathcal{L}_{\text{Task}}$ and $\mathcal{L}_{\text{Meta}}$ through the back-propagation strategy. The complete training process is divided into two processes, i.e., $\mathcal{L}_{\text{Task}}$ and $\mathcal{L}_{\text{Meta}}$ optimization. The Algorithm 1 outlines the training process of our architecture. Specifically, there are two types of trainable parameters. Let ω_{mu} and ω_{bk} indicate the trainable parameters of the MuGC and the Backbone Network, respectively. Let ω_{mt} indicate the trainable parameters in the Meta Learner.

In $\mathcal{L}_{\text{Task}}$ optimization, with Meta Learner fixed and given the receptive field, we update ω_{mu} and ω_{bk} by gradient descent (lines 3–5 of Algorithm 1).

Similarly, in the $\mathcal{L}_{\text{Meta}}$ optimization (lines 6–9 of Algorithm 1), we first randomly select the receptive field of instances to explore the task loss of different receptive fields. For MuGC and Backbone Network fixed and given the receptive field, we then calculate $\mathcal{L}_{\text{Task}}$. Finally, we update ω_{mt} to accurately predict $\mathcal{L}_{\text{Task}}$ and the maximum score is the best receptive field.

4.5. Time complexity analysis

In this subsection, we provide a theoretical analysis of the time complexity of our architecture. Existing graph neural networks can be broadly categorized into three types based on their aggregation methods: neighbor-based GNNs (e.g., GCN and GAT), path-based GNNs (e.g., GraphSAGE), and subgraph-based GNNs (e.g., ADRP-GNN). The time complexity of different methods in the same category may vary due to aggregate manners.

Above all, we define a graph with m nodes, where each node has n neighbors, and the model's receptive field is k. Following the general paradigm of GNNs, the node representations are first mapped, then aggregated, and finally, the target node representation is updated. The time complexity of aggregating a node to a target node is $O(1)$, assuming no complex design. We analyze the time complexity for inferring a node (target node) in each of the three categories.

Neighbor-based GNNs: This method performs message passing between pairs of nodes connected by edges, and this is repeated k times. Therefore, the time complexity is $O(kmn)$, where m is the number of nodes, n is the number of neighbors per node, and k is the number of hops.

Path-based GNNs: In these approaches, neighbors are sampled in terms of paths, with the constant sampling rate. The message passing is performed over the sampled edges, and this operation is repeated k times. As the sampling rate is a constant and does not affect the time complexity, the time complexity is still $O(kmn)$.

Subgraph-based GNNs: In this type, the target node aggregates all neighbors up to k-hops. The time complexities of MuGC and Meta Learner are $O(1)$, which are similar to traditional GNNs. The number of neighbors grows exponentially as $\sum_i^k n^i$, and thus the time complexity of subgraph-based methods such as ADRP-GNN is $O(n^k)$. (In

Table 1
Details of the datasets.

Dataset	Nodes	Edges	Degree	Features	Classes
Cora	2708	5429	2.00	1433	7
CiteSeer	3327	4732	1.42	3703	6
PubMed	19,717	44,338	2.25	500	3
Facebook	22,470	342,004	7.61	128	4
Actor	7600	30,019	7.90	932	5
Cornell	183	298	3.26	1703	5
Texas	183	325	3.55	1703	5
Wisconsin	251	515	4.10	1703	5

case of stacked convolutional layers, the time complexity is $O(kmn)$.) Therefore, the primary factor affecting the time complexity in ADRP-GNN is the aggregation of the large number of neighbors. The time consumption increases significantly for high-degree nodes on large graphs, e.g., social networks.

In the case of Meta Learner, its presence does not necessarily increase the computational cost. For instance, when the receptive field is set to 10, traditional GNNs would require 10 rounds of message passing. In contrast, Meta Learner might terminate the message passing early, effectively reducing the computational cost.

Intuitively, an aggressive solution is to downsample neighbors of each hop to decrease the time complexity of ADRP-GNN. So, we propose a strategy that significantly reduces computational complexity and accelerates computation. Specifically, we sample $n_i = p_i nm$ nodes from k_i-hop neighbors, where p_i is the constant sampling rate. The number of nodes that target nodes aggregate is $\sum_i^k n_i$. Theoretically, our architecture and traditional GNNs aggregate an exponential and linear number of neighbors, respectively. Sampling neighbors of receptive fields reduces the time complexity to linear, which is the same as that of traditional GNNs. Therefore, the time complexity of our framework decreases to $O(knm)$.

It is important to note that this does not imply that the computation time of MuGC after sampling is necessarily lower than that of traditional methods in practical applications. This is mainly due to two reasons: (1) the sampling ratio affects the actual number of aggregated neighbors, and (2) when inferring all nodes simultaneously, intermediate results in traditional methods can be reused, significantly reducing the overall computation time.

In addition, the architecture in terms of time complexity is advantageous compared to traditional GNNs with the same receptive field, as it has only one layer and does not frequently transform and aggregate messages. Summarily, the time computation is acceptable in scenarios, and the time consumption is analyzed in the following experiments.

5. Experiments

In this section, we set up multiple experiments to evaluate the effectiveness of ADRP-GNN on real-world tasks. We perform node classification to illustrate the ability of ADRP-GNN. We also conduct an over-smoothing analysis to further prove the ADRP-GNN solution and examine time consumption. In addition, we demonstrate the effects of each component through an ablation study.

5.1. Datasets

We conduct experiments on eight real-world datasets and the details of the datasets are shown in Table 1.

- Citation Network. In *Cora*,[1] *CiteSeer*, and *PubMed*, nodes, edges, and features of the citation network correspond to papers, paper citation relations, and the content of papers, respectively. For node classification, node labels are research fields.
- Social Network. In *Facebook*,[2] it consists of a social network of Facebook pages where nodes represent pages and edges represent mutual likes between them. For the node classification task, labels have four categories: politicians, organizations, television shows, and companies. *Actor*[3] is a co-occurrence network where nodes represent actors and edges represent co-appearances in movies. Each node has a feature vector based on the genres of the movies in which the actor has appeared. This could be a binary vector where each dimension represents a specific genre.
- Webpage Network. *Cornell*,[4] *Texas*, and *Wisconsin* are subsets of the WebKB dataset, whose nodes represent web pages, and edges are hyperlinks between them. Node features are the bag-of-words representation of web pages. The web pages are manually classified into the five categories, student, project, course, staff, and faculty.

5.2. Baselines

We compare our architecture with several strong baselines.

For traditional GNN methods, GraphSAGE adopts a fixed-size neighbor sampling strategy to construct regular data and expand receptive fields. GCN, which lies in the spectral domain, heavily relies on the graph Laplacian matrix to guide aggregate neighbors to target nodes. GAT, which lies in the spatial domain, utilizes an attention mechanism to combine neighboring features to target nodes. GIN (Xu et al., 2019) is a robust GNN framework to capture different graph structures with the same ability as the Weisfeiler-Lehman graph isomorphism test. JK-Net (Xu et al., 2018) leverages skip connections to go deeper for larger receptive fields. We report the best results from the multiple architectures of JK-Net. NLGCN (Liu, Wang, & Ji, 2021) proposes an effective non-local aggregation framework with an efficient attention-guided sorting for GNNs.

For recent GNN methods solving over-smoothing, GeniePath (Liu et al., 2019) obtains adaptive receptive fields by both breadth and depth exploration. DropEdge (Rong et al., 2019) randomly removes a certain number of edges from the input graph at each training epoch, acting like a data augmenter and also a message-passing reducer. GCNII (Chen, Wei et al., 2020) utilizes Initial residuals and Identity mapping. DAGNN (Liu et al., 2020) adaptively incorporates information from large receptive fields. SMEGCN (Jiang et al., 2022) introduces the local topology into the graph Laplacian by introducing the notion of the motif. SJLR (Giraldo et al., 2023) introduces the Stochastic Jost and Liu Curvature Rewiring algorithm, which performs edge addition and removal during GNN training while maintaining the graph unchanged during testing. GraphSAGE++ (Jiawei et al., 2024) extracts the representation of the target node at each layer and then concatenates all layer weighted representations to obtain the final result. NASC (Sancak, Balin, & Catalyurek, 2024) proposes a novel skip connection with an adaptive weighting strategy.

5.3. Experimental settings

We implement our proposed framework using PyTorch 2.0.1 and PyTorch Geometric 2.3.0 and train the model on a server with NVIDIA RTX4090 GPU. We use the following sets of hyperparameters for the above benchmark datasets: 64 (number of hidden units) for MuGC, 64 (number of hidden units) for Backbone Network, and 64 (number of hidden units) for Meta Learner. The batch size is set to 1024, and the model is trained for 200 epochs. We use the Adam optimizer with a learning rate of 0.005 and a weight decay of 0.0001. The receptive field is set to 10. For the compared baselines, we closely follow their

[1] https://linqs.soe.ucsc.edu/datasets

[2] http://snap.stanford.edu/data

[3] http://en.wikipedia.org/wiki/Category:English-language_films

[4] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb

Table 2
Performance results on Cora, CiteSeer, PubMed, and Facebook in terms of node classification accuracy (in percent). The best results are boldfaced. The underlined numbers are the second best results.

Dataset	Cora	CiteSeer	PubMed	Facebook
GraphSAGE	88.17 ± 1.95	77.11 ± 1.78	84.81 ± 0.62	89.92 ± 0.58
GCN	87.12 ± 1.62	76.73 ± 1.45	87.58 ± 0.66	91.61 ± 0.36
GAT	87.48 ± 1.25	76.58 ± 1.51	86.05 ± 0.54	90.38 ± 0.49
GIN	81.73 ± 1.82	68.46 ± 1.90	85.57 ± 0.59	90.03 ± 0.57
JK-Net	86.61 ± 0.93	75.57 ± 1.23	86.80 ± 0.42	91.84 ± 0.28
NLGCN	87.80 ± 1.83	77.63 ± 1.33	87.55 ± 0.67	90.96 ± 0.34
GeniePath	85.98 ± 2.01	74.62 ± 0.94	88.31 ± 0.44	89.03 ± 0.36
DropEdge	86.34 ± 1.18	74.87 ± 1.92	86.15 ± 0.59	91.73 ± 0.29
GCNII	87.31 ± 1.34	77.13 ± 1.93	88.46 ± 2.01	91.97 ± 2.08
DAGNN	88.29 ± 1.35	77.67 ± 1.44	87.81 ± 0.52	91.34 ± 0.76
SMEGCN	88.74 ± 1.26	78.28 ± 1.86	88.16 ± 0.46	90.82 ± 1.21
SJLR	88.73 ± 1.37	76.35 ± 1.82	88.56 ± 0.68	89.24 ± 1.42
GraphSAGE++	80.77 ± 1.73	71.81 ± 1.90	86.15 ± 0.64	87.41 ± 1.58
NASC	87.44 ± 1.54	77.02 ± 2.11	87.93 ± 0.97	91.28 ± 1.79
ADRP-GNN	**89.48 ± 1.78**	**79.83 ± 2.04**	**89.02 ± 0.81**	**92.55 ± 1.83**
△	+0.83%	+1.94%	+0.52%	+0.63%

Table 3
Performance results on Actor, Cornell, Texas, and Wisconsin in terms of node classification accuracy (in percent). The best results are boldfaced. The underlined numbers are the second best results.

Dataset	Actor	Cornell	Texas	Wisconsin
GraphSAGE	30.37 ± 1.37	47.32 ± 4.76	54.79 ± 6.46	55.33 ± 6.28
GCN	27.43 ± 1.04	45.95 ± 4.81	55.26 ± 2.24	50.98 ± 5.08
GAT	28.61 ± 1.02	48.64 ± 2.79	55.56 ± 4.48	54.90 ± 7.03
GIN	25.39 ± 0.72	45.95 ± 4.56	55.26 ± 5.61	47.06 ± 3.36
JK-Net	28.29 ± 1.12	54.05 ± 4.63	56.02 ± 5.85	49.01 ± 6.02
NLGCN	32.38 ± 1.93	58.12 ± 5.43	66.59 ± 6.17	58.31 ± 7.27
GeniePath	32.26 ± 0.93	55.86 ± 5.13	57.78 ± 6.33	58.82 ± 5.56
DropEdge	24.86 ± 0.82	56.75 ± 8.38	34.21 ± 7.07	49.01 ± 7.41
GCNII	29.39 ± 1.97	58.28 ± 5.17	62.17 ± 6.05	58.83 ± 3.72
DAGNN	32.97 ± 1.17	56.24 ± 6.91	64.63 ± 3.72	49.01 ± 4.45
SMEGCN	31.58 ± 1.87	60.40 ± 4.21	68.41 ± 3.16	64.29 ± 5.09
SJLR	30.79 ± 1.45	48.64 ± 5.88	53.51 ± 4.14	58.19 ± 4.72
GraphSAGE++	28.79 ± 1.62	46.39 ± 6.26	52.60 ± 4.66	52.91 ± 5.13
NASC	30.89 ± 1.74	59.42 ± 7.85	53.96 ± 5.64	60.69 ± 5.49
ADRP-GNN	**34.54 ± 1.61**	**64.86 ± 5.24**	**70.27 ± 5.62**	**68.63 ± 5.35**
△	+4.55%	+6.88%	+2.65%	+6.32%

settings reported in the relevant papers. We utilize the Adam optimizer with an initial learning rate of 0.01 for the task-loss function and 0.01 for the meta-loss function, respectively. In all cases, we use an early stopping strategy on the validation datasets. For all graph datasets, we randomly split nodes of each class into 60%, 20%, and 20% for training, validation, and testing, which is the same setting as in Pei, Wei, Chang, Lei, and Yang (2019). We repeat the experiments 10 times and report the average results and standard deviation.

5.4. Performance results

The performances of the competing baselines and ADRP-GNN are shown in Tables 2 and 3. The methods are mainly divided into two types: (1) Traditional GNNs in the setting of a reasonable receptive field can achieve a similar performance with other methods. (2) The existing GNN methods utilize various ways to overcome the over-smoothing issue. The methods learn deep and distinguishable representations, which is an advantage over other baselines. Compared to the baselines, ADRP-GNN has a larger receptive field at the same or fewer number of layers and parameters. In addition, ADRP-GNN utilizes Meta Learner to select suitable receptive fields, which help the node obtain sufficient structural information and avoid introducing redundant information. However, due to the complicated structure of the Meta Learner and the lack of the ground truth value for its task, which results in training difficulties and large standard deviations. The results show that ADRP-GNN outperforms all the baselines in accuracy for node classification, achieving 0.52% to 6.88% improvements on these datasets.

5.5. Over-smoothing analysis

To evaluate the performance of ADRP-GNN on the over-smoothing issue, we conduct experiments from two aspects: whether the over-smoothing issue occurs when MuGC increases the receptive field, and whether Meta Learner can select the suitable receptive field.

For the first aspect, we drop Meta Learner from ADRP-GNN and manually set the MuGC's receptive fields. We choose the traditional GNN (GCN and GAT) as a comparison for the performance of the model under different receptive fields. The traditional GNN enlarges the receptive field by stacking graph convolutional layers. As shown in Figs. 4(a) and (c), traditional GNN experiences the over-smoothing issue, especially when the depth is greater than 32, the performance degrades significantly, while performances of MuGC and the model (GeniePath and DAGCN) for over-smoothing fluctuate only slightly as the receptive field increases. In summary, the experimental results highlight that while traditional GNNs suffer from significant performance degradation due to over-smoothing as the receptive field increases, MuGC, along with the model for over-smoothing, maintain stable performance despite the expansion of the receptive field.

To rule out the possibility that the performance degradation is due to model complexity causing overfitting, we present Figs. 4(b) and (d) for comparison. As the receptive field increases, the models' performance declines on both the training and test sets, suggesting that the performance degradation is not attributable to overfitting.

Furthermore, to offer a more rigorous evaluation of over-smoothing, we introduce the Dirichlet energy metric, which provides a quantitative measure of the smoothness in the learned node representations.

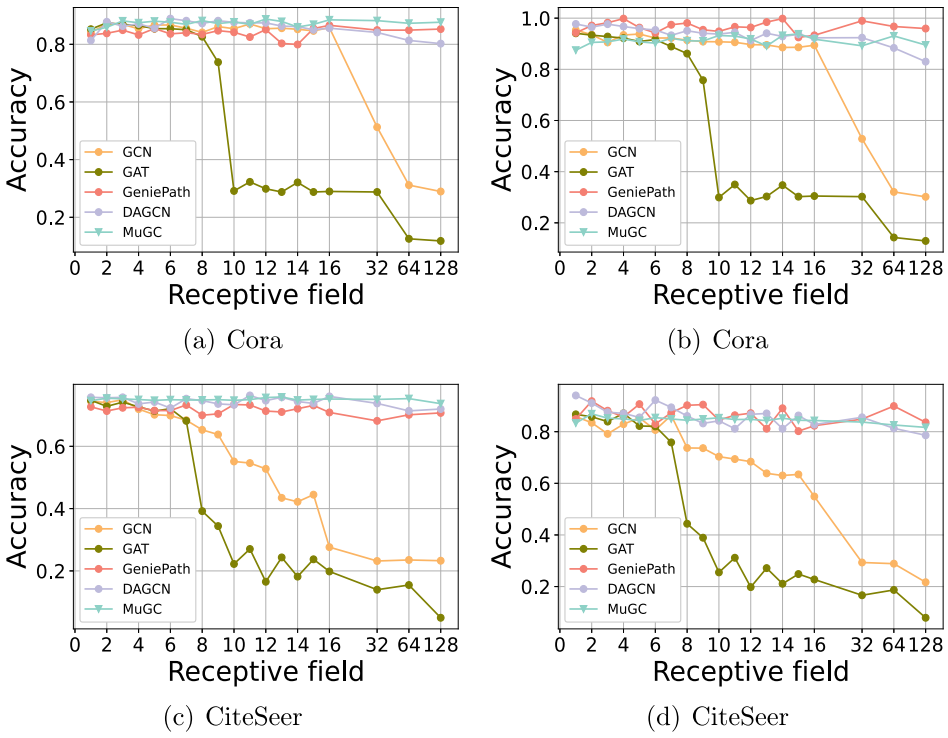


Fig. 4. (a) and (c) are the classification performances on the test sets. (b) and (d) are the classification performances on the training sets.

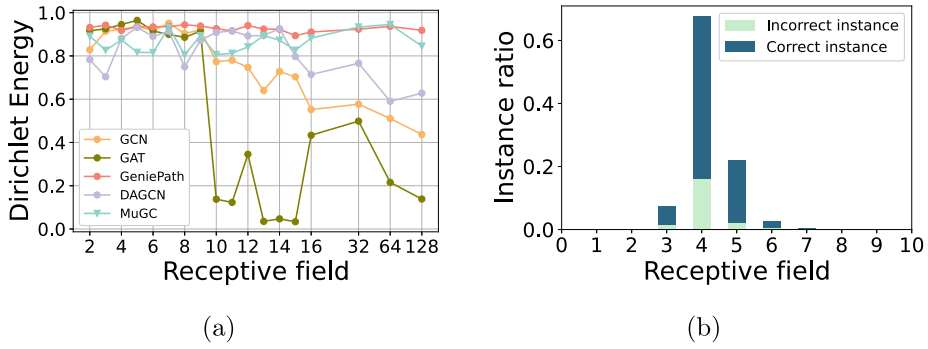


Fig. 5. (a) Dirichlet Energy with respect to the receptive field on CiteSeer. (b) The ratio of correct and incorrect classifications under the receptive fields which Meta Learner predicts on CiteSeer.

Fig. 5(a) displays the Dirichlet energy across various receptive fields. As the receptive field increases, the Dirichlet energy of traditional GNNs decreases, indicating that the node representations become more similar to one another. In contrast, the models designed to address over-smoothing maintain largely stable Dirichlet energy values. In summary, the Dirichlet energy analysis shows that traditional GNNs experience over-smoothing with increasing receptive field, while models addressing over-smoothing maintain stable values.

To investigate our framework, we visualize the embedding values of MuGC and GCN. As illustrated in Fig. 6, the embedding of different category nodes is distinguishable when the depth of MuGC and GCN is 4. Meanwhile, the embedding of our framework has no significant changes comparing depths of 4 and 64, which demonstrates the over-smoothing issue relieved. However, the embedding of GCN becomes indistinguishable with the depth increasing to 64, that is, the over-smoothing issue. The fluctuation is due to the noise introduced by the increased receptive fields, so it is necessary to select a suitable receptive field.

For the other aspect, we set the maximum depth of the MuGC as 10 and count the correct and incorrect instance ratios that ADRP-GNN predict under the receptive fields of Meta Learner output, which is shown in Fig. 5(b). The optimal receptive fields of nodes are 3 to 6, and the traditional GNNs exhibit excellent performance in this range of receptive fields, so the receptive fields that Meta Learner predicts are consistent with the number of layers of the traditional GNN. Although 2-hop structural information for some nodes is sufficient to address the classification task, node information of larger receptive fields is beneficial for node representations, so the Meta Learner evaluates higher scores and selects larger receptive fields. In the task, neighbors at distances greater than 6 have relatively little effect or introduce noise on the target node representations.

5.6. Time consumption analysis

We further examine the time consumption of the architecture on the large dataset. The critical difference in the time consumption of ADRP-GNN is the MuGC. We assess the effect of the number of samplings on the ability where the MuGC learns structure information and the time consumption.

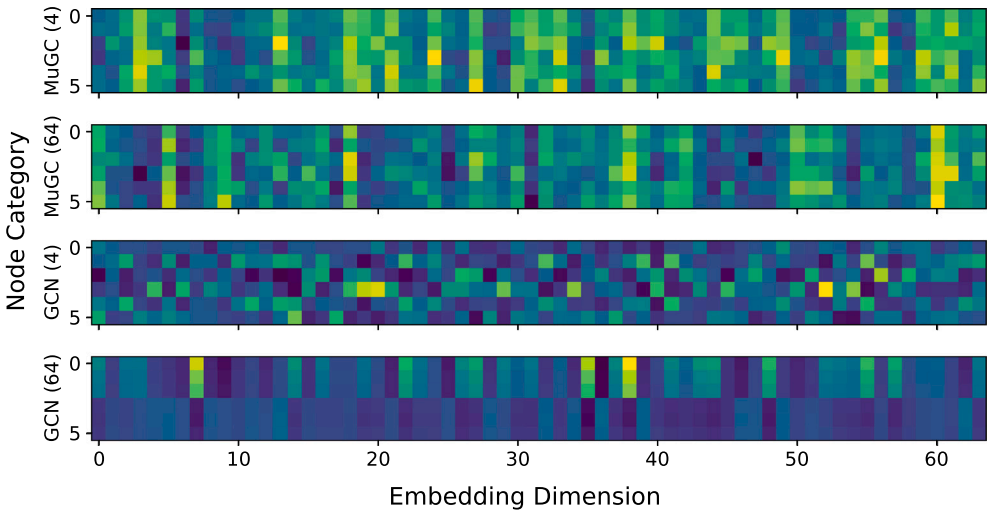


Fig. 6. Embedding visualizations on CiteSeer. It shows the embedding values of MuGC and GCN for stacked layers of 4 and 64, respectively. The colors denote the embedding values, where the brighter the color, the larger the value. The embedding dimension is 64, as in the horizontal axis. As shown in the vertical axis, we randomly select one node in each category, for a total of 6 nodes.

Table 4
Time consumption analysis of MuGC with a sample strategy on Facebook. The "# sample nodes" indicates the number of sample neighbors from each receptive field. For convenience, the number of nodes sampled by the first hop is the average degree, and the other hops sample multiples of the average degree. We set the depth of the GCN and GAT to be the same as the receptive field of MuGC.

Method	# Sample nodes	Acc.	Computation time (s)	
			Training	Inference
MuGC	1/2	88.53	0.114	0.058
	1	89.86	0.163	0.079
	2	90.62	0.257	0.135
	4	91.37	0.276	0.183
	all	92.56	0.326	0.245
GCN	–	54.64	0.047	0.026
GAT	–	33.18	0.090	0.038

Specifically, for the sampling strategy, we randomly sample the number based on the average degree of a graph, which is 8 on Facebook. We record the computation time of training and inference, where the former is the time of an epoch and the latter is the predicted test dataset. We adopt traditional models with the same depth as compared.

As shown in Table 4, as the number of sampling nodes increases, the MuGC gets better performance but consumes more time. Theoretically, the time complexity of MuGC is the same as that of traditional GNNs when all hops sample 8 nodes, but the MuGC consumes more time than traditional GNNs because of engineering realization. The performance and time consumption are better balanced when the MuGC samples fourfold nodes. A better sampling strategy and the number of sampling nodes need to be further explored.

5.7. Ablation study

This subsection describes ablation studies of ADRP-GNN in Table 5 to validate the effectiveness of key components. We can observe that: (1) When we drop the Backbone Network, comparing the experimental results in rows 1, 2 and rows 3, 4 of Table 5, respectively, the performance of our architecture declines. It demonstrates the necessity of the Backbone Network increasing architecture depth. (2) To evaluate the effect of MuGC, we drop the component (row 5 in Table 5) or replace it with traditional GNN (GAT) (row 6 in Table 5), and the result illustrates that MuGC enlarges the receptive field to benefit architecture performance. (3) We add Meta Learner to the architecture, which consists of MuGC and the Backbone Network, and discover that the performance of the architecture is an enhancement, comparing rows 1 and 3 in Table 5.

Table 5
Ablation study of architecture on CiteSeer.

Accuracy	GAT	MuGC	Meta learner	Backbone network
79.83		√	√	√
73.24		√	√	
77.97		√		√
70.41		√		
58.31				√
75.28	√			√

6. Conclusion

In this paper, we tackle the over-smoothing issue that limits the model depth in the GNN field. We propose a novel architecture that adaptively constructs receptive fields, and aggregates messages by a monolayer graph convolution layer, to relieve the over-smoothing issue. Our method proposes a new perspective decoupling the number of stacked graph convolution layers and the receptive field into two structures, i.e., MuGC and Backbone Network, where the former enlarges receptive fields with a monolayer graph convolution layer and the latter increases the model depth to enhance learning ability. To avoid noise introduced by larger receptive fields, we present Meta Learner to select suitable receptive fields for nodes. Our study reveals a previously unexplored way that monolayer fuses distant neighbor information and variable receptive fields, offering new insights into over-smoothing representations. On a real-world dataset, the proposed approach achieves competitive results and better solves the over-smoothing issue. In the future, we will extend our framework to investigate the usage of spatio-temporal representation, heterogeneous graph neural networks, and unsupervised tasks.

CRediT authorship contribution statement

Hepeng Gao: Writing – review & editing, Writing – original draft, Methodology. **Funing Yang:** Supervision. **Yongjian Yang:** Resources. **Yuanbo Xu:** Writing – review & editing. **Yijun Su:** Writing – review & editing.

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.

Acknowledgments

This work was supported by the Science and Technology Development Program of Jilin Province (No. 20240302093GX) and the National Natural Science Foundation of China (No. 62072209).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2025.107658.

Data availability

I have provided access links (https://github.com/wn13/ADRPGNN) to the code and data related to the experiments in the paper.

References

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2020). N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *Uncertainty in artificial intelligence* (pp. 841–851). PMLR.

Bo, D., Wang, X., Shi, C., & Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3950–3957).

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3438–3445).

Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020). Simple and deep graph convolutional networks. In *Proceedings of machine learning research: Vol. 119, Proceedings of the 37th international conference on machine learning* (pp. 1725–1735).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems* (pp. 3837–3845).

Donnat, C., & Jeong, S. W. (2023). Studying the effect of GNN spatial convolutions on the embedding space's geometry. In *Uncertainty in artificial intelligence* (pp. 539–548). PMLR.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., et al. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems*, *28*.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., et al. (2020). Graph random neural networks for semi-supervised learning on graphs. In *Advances in neural information processing systems*.

Giraldo, J. H., Skianis, K., Bouwmans, T., & Malliaros, F. D. (2023). On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM international conference on information and knowledge management* (pp. 566–576). ACM.

Gregucci, C., Nayyeri, M., Hernández, D., & Staab, S. (2023). Link prediction with attention applied on multiple knowledge graph embedding models. In *Proceedings of the ACM web conference 2023* (pp. 2600–2610).

Guo, H., & Mao, Y. (2023). Interpolating graph pair to regularize graph classification. In *Proceedings of the AAAI conference on artificial intelligence: Vol. 37* (pp. 7766–7774).

Hu, R., Pan, S., Long, G., Lu, Q., Zhu, L., & Jiang, J. (2020). Going deep: Graph convolutional ladder-shape networks. In *Proceedings of the AAAI conference on artificial intelligence*.

Jiang, X., Ji, P., & Li, S. (2019). CensNet: Convolution with edge-node switching in graph neural networks. In *IJCAI* (pp. 2656–2662).

Jiang, X., Yang, Z., Wen, P., Su, L., & Huang, Q. (2022). A sparse-motif ensemble graph convolutional network against over-smoothing. In *Proc. 31st int. joint conf. artif. intell.* (pp. 2094–2100).

Jiawei, E., Zhang, Y., Yang, S., Wang, H., Xia, X., & Xu, X. (2024). GraphSAGE++: Weighted multi-scale GNN for graph representation learning. *Neural Processing Letters*, *56*(1), 24.

Ju, W., Mao, Z., Yi, S., Qin, Y., Gu, Y., Xiao, Z., et al. (2024). Hypergraph-enhanced dual semi-supervised graph classification. In *International conference on machine learning* (pp. 22594–22604). PMLR.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Li, Q., Han, Z., & Wu, X. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3538–3545).

Li, J., Shomer, H., Mao, H., Zeng, S., Ma, Y., Shah, N., et al. (2023). Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *Advances in Neural Information Processing Systems*, *36*, 3853–3866.

Li, R., Wang, S., Zhu, F., & Huang, J. (2018). Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3546–3553).

Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., et al. (2019). GeniePath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4424–4431).

Liu, M., Gao, H., & Ji, S. (2020). Towards deeper graph neural networks. In *The 26th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 338–348).

Liu, X., Li, X., Fiumara, G., & De Meo, P. (2023). Link prediction approach combined graph neural network with capsule network. *Expert Systems with Applications*, *212*, Article 118737.

Liu, Y., Liang, K., Xia, J., Zhou, S., Yang, X., Liu, X., et al. (2023). Dink-net: Neural clustering on large graphs. In *International conference on machine learning* (pp. 21794–21812). PMLR.

Liu, M., Wang, Z., & Ji, S. (2021). Non-local graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(12), 10270–10276.

Liu, Y., Yang, X., Zhou, S., Liu, X., Wang, S., Liang, K., et al. (2023). Simple contrastive graph clustering. *IEEE Transactions on Neural Networks and Learning Systems*.

Liu, C., Zhan, Y., Ma, X., Ding, L., Tao, D., Wu, J., et al. (2023). Gapformer: Graph transformer with graph pooling for node classification. In *IJCAI* (pp. 2196–2205).

Liu, Y., Zhou, C., Pan, S., Wu, J., Li, Z., Chen, H., et al. (2023). CurvDrop: A Ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In *Proceedings of the ACM web conference 2023* (pp. 221–230).

Lu, K., Yu, Y., Fei, H., Li, X., Yang, Z., Guo, Z., et al. (2024). Improving expressive power of spectral graph neural networks with eigenvalue correction. In *Proceedings of the AAAI conference on artificial intelligence: Vol. 38* (pp. 14158–14166).

Luan, S., Hua, C., Xu, M., Lu, Q., Zhu, J., Chang, X.-W., et al. (2023). When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. *Advances in Neural Information Processing Systems*, *36*, 28748–28760.

Luo, Y., Shi, L., & Wu, X.-M. (2025). Unlocking the potential of classic GNNs for graph-level tasks: Simple architectures meet excellence. arXiv preprint arXiv:2502.09263.

Ma, G., Hu, C., Ge, L., & Zhang, H. (2023). Multi-view robust graph representation learning for graph classification. In *IJCAI* (pp. 4037–4045).

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, P. K., Coates, M., et al. (2023). Graph inductive biases in transformers without message passing. In *International conference on machine learning* (pp. 23321–23337). PMLR.

Ma, X., Wang, J., Chen, H., & Song, G. (2021). Improving graph neural networks with structural adaptive receptive fields. In *Proceedings of the web conference* (pp. 2438–2447).

Monti, F., Bronstein, M., & Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. *Advances in Neural Information Processing Systems*, *30*.

Pan, E., & Kang, Z. (2023). Beyond homophily: Reconstructing structure for graph-agnostic clustering. In *International conference on machine learning* (pp. 26868–26877). PMLR.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., & Yang, B. (2019). Geom-GCN: Geometric graph convolutional networks. In *International conference on learning representations*.

Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). DropEdge: Towards deep graph convolutional networks on node classification. In *International conference on learning representations*.

Sancak, K., Balin, M. F., & Catalyurek, U. (2024). Do we really need complicated graph learning models? – a simple but effective baseline. In *The third learning on graphs conference*.

Tan, Z., Guo, R., Ding, K., & Liu, H. (2023). Virtual node tuning for few-shot node classification. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2177–2188).

Tan, Q., Zhang, X., Liu, N., Zha, D., Li, L., Chen, R., et al. (2023). Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In *Proceedings of the sixteenth ACM international conference on web search and data mining* (pp. 625–633).

Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2023). Graph clustering with graph neural networks. *Journal of Machine Learning Research*, *24*(127), 1–21.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations*.

Wang, Z., Di, S., & Chen, L. (2023). A message passing neural network space for better capturing data-dependent receptive fields. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2489–2501).

Wang, Y., Yi, K., Liu, X., Wang, Y. G., & Jin, S. (2023). ACMP: Allen-Cahn message passing with attractive and repulsive forces for graph neural networks. In *The eleventh international conference on learning representations*. OpenReview.net.

Wang, X., & Zhang, M. (2022). How powerful are spectral graph neural networks. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of machine learning research*: *Vol. 162*, *Proceedings of the 39th international conference on machine learning* (pp. 23341–23362). PMLR.

Xhonneux, L.-P., Qu, M., & Tang, J. (2020). Continuous graph neural networks. In *Proceedings of machine learning research*: *Vol. 119*, *Proceedings of the 37th international conference on machine learning* (pp. 10432–10441).

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *International conference on learning representations*.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *Proceedings of machine learning research*: *Vol. 80*, *Proceedings of the 35th international conference on machine learning* (pp. 5453–5462).

Yang, M., Shen, Y., Li, R., Qi, H., Zhang, Q., & Yin, B. (2022). A new perspective on the effects of spectrum in graph neural networks. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of machine learning research*: *Vol. 162*, *Proceedings of the 39th international conference on machine learning* (pp. 25261–25279). PMLR.

Yang, L., Wang, C., Gu, J., Cao, X., & Niu, B. (2021). Why do attributes propagate in graph convolutional neural networks? In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4590–4598).

Yin, N., Shen, L., Wang, M., Luo, X., Luo, Z., & Tao, D. (2023). OMG: Towards effective graph classification against label noise. *IEEE Transactions on Knowledge and Data Engineering*, *35*(12), 12873–12886.

Zhang, G., Cheng, D., Yuan, G., & Zhang, S. (2024). Learning fair representations via rebalancing graph structure. *Information Processing & Management*, *61*(1), Article 103570.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D. (2018). GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the thirty-fourth conference on uncertainty in artificial intelligence* (pp. 339–349).

Zhang, S., Zhang, J., Song, X., Adeshina, S., Zheng, D., Faloutsos, C., et al. (2023). PaGE-Link: Path-based graph neural network explanation for heterogeneous link prediction. In *Proceedings of the ACM web conference 2023* (pp. 3784–3793).

Zhang, K., Zhu, Y., Wang, J., & Zhang, J. (2020). Adaptive structural fingerprints for graph attention networks. In *International conference on learning representations*.

Zhao, J., Jin, D., Ge, M., Shan, L., Wang, X., He, D., et al. (2024). Fug: Feature-universal graph contrastive pre-training for graphs with diverse node features. *Advances in Neural Information Processing Systems*, *37*, 4003–4034.

Zhao, S., Zheng, Y., Li, J., Zhang, X., Tang, C., & Tan, Z. (2024). Pure kernel graph fusion tensor subspace clustering under non-negative matrix factorization framework. *Information Processing & Management*, *61*(2), Article 103603.