A Message Passing Neural Network Space for Better Capturing Data-dependent Receptive Fields

Zhili Wang
HKUST
Hong Kong SAR, China
zwangeo@connect.ust.hk

Shimin Di*
HKUST
Hong Kong SAR, China
sdiaa@connect.ust.hk

Lei Chen
HKUST (GZ), Guangzhou, China
HKUST, Hong Kong SAR, China
leichen@cse.ust.hk

ABSTRACT

Recently, the message passing neural network (MPNN) has attracted a lot of attention, which learns node representations based on the receptive field of the given node. Despite its success in many graph-related tasks, recent studies find that conventional MPNNs are incapable of handling variant receptive fields required in different graphs, and thereby some upgraded MPNNs have been developed. However, these methods are limited to designing a common solution for different graphs, which fails to capture the impact of different graph properties on the receptive fields. To alleviate such issues, we propose a novel MPNN space for data-dependent receptive fields (MpnnDRF), which enables us to dynamically design suitable MPNNs to capture the receptive field for the given graph. More concretely, we systemically investigate the capability of existing designs and propose several key design dimensions to improve them. Then, to fully explore the proposed designs and useful designs in existing works, we propose a novel search space to incorporate them and formulate a search framework. In the empirical study, the proposed MpnnDRF shows very strong robustness against the increased receptive field, which allows MpnnDRF to learn node representations based on a larger perceptual field. Therefore, MpnnDRF consistently achieves outstanding performance on benchmark node and graph classification tasks.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**; • **Networks** → **Network design principles**.

KEYWORDS

Message Passing Neural Network, Over-Smoothing, Over-Squashing

ACM Reference Format:
Zhili Wang, Shimin Di, and Lei Chen. 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 (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599243

*Shimin DI is the corresponding author.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599243

1 INTRODUCTION

As one of the most ubiquitous data structures, graph is adopted to model various complex real-world systems, e.g., recommendation systems [25, 37], physical systems [54], biochemical networks [20], and knowledge graphs [12, 28]. To handle different graph scenarios, graph representation learning encodes the graph into the low dimensional vector space, which has shown promising results on various graph tasks, e.g., node classification [30], link prediction [76], and graph classification [72]. Recently, message passing neural networks (MPNNs) [1, 11, 23, 24, 30, 31, 38, 50, 63, 70, 72, 73] have become the leading approaches on graph representation learning. MPNNs iteratively aggregate and transform messages from neighbors to update node representations. Conventional MPNNs rely on receptive field of given node to gather information for representation learning. Given one target node, the receptive field denotes a set of nodes in the graph which influence the final representation of the given node. Generally, the receptive field is affected by two factors: 1) the mechanism of how receptive field expands at each iteration(layer), 2) how many iterations K of expansion are needed. Fig. 1 illustrates an example that 1*hop* mechanism with 2 layers.

Despite the success, general MPNNs may be incapable of handling variant receptive fields required in different graph data sets. First, various graphs may require different MPNN layers K for appropriate receptive field size. For those graphs with low density (e.g., biological molecule), the receptive field expand slowly. Thus, deep layers (large K) can help MPNNs gather sufficient information from the K-hop neighbors of target nodes. But MPNNs with deep layers may make the node representations indistinguishable (a.k.a. over-smoothing issue [40, 42, 47, 73]) on dense graphs (e.g., social networks). Instead, shallow MPNN layers will be more suitable for dense cases. Second, various graphs may require different MPNN mechanisms to achieve more descriptive receptive field. On some homophily graphs where features or labels of nodes tend to be similar (e.g., citation and social networks), the 1*hop* mechanism in conventional MPNNs is reasonable, because using local surrounding regions as receptive field is informative enough for center node. On the contrary, for those heterophily graphs where features or labels of nodes tend to be dissimilar (e.g., protein networks), a much larger receptive field is needed in the 1*hop* mechanism to aware information from distant nodes. But it may cause the over-squashing issue [2, 62] that useful messages from distant nodes are severely distorted due to the exponentially expanded size of receptive field.

Recent works explore some of these issues and propose solutions. Node-adaptive MPNNs [42, 78, 80] observe inconsistent receptive field expansion speeds and smoothing levels within one graph caused by diverse node local patterns and substructures. To alleviate over-smoothing, they adopt different layers for each node

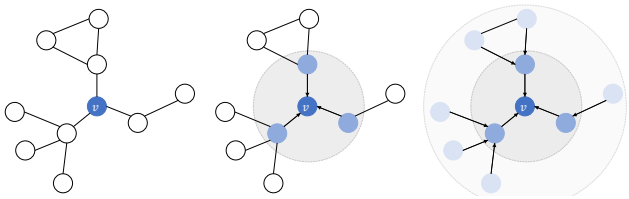


Figure 1: The illustration to MPNN and receptive field.

to better suit their actual demands of receptive fields. Decoupled MPNNs [6, 8, 31, 42, 53, 58, 70, 78] propose that deep layers and large receptive fields may not necessarily cause over-smoothing. Instead, they argue the coupled relationship between aggregation and transformation. Thus, they develop decoupled approaches to combine shallow transformation layers with deep aggregation layers. Moreover, attentive methods [3, 4, 16, 17, 34, 45, 63, 65] design weighed receptive field that better balance node importance. Graph re-wiring approaches [2, 34, 43, 50, 59, 62, 66, 83] go beyond the conventional 1*hop* mechanism for better receptive field.

Although these works alleviate inconsistent receptive field requirements in different graph data sets, there are still some obvious problems exist. First, existing works cannot consider the impact of different graph properties on receptive field in a balanced way. For example, some works are designed to handle over-smoothing, but we conclude that they may only work on the homophily graph due to over-squashing through empirical study (see Sec. 3.2). In other words, many works often focus on tackling one issue while ignoring other problems. Second, the designs of mainstream solutions to heterophily scenarios or over-squashing can be further improved. For instance, multi-channel MPNNs still partially rely on the 1*hop* neighbor mechanism to aggregate information, which may be harmful in some cases (see more in Sec. 3.2). Besides, how to learn from different channels in multi-channel MPNNs also requires carefully design. Fully-connected MPNNs raise the concern of higher model complexities. Moreover, many of them still follow the coupled way to aggregate and transform information. But such coupled design has been demonstrated unnecessary and may even hinder performances [31, 42, 70]. Third, these works attempt to provide a common solution for different graph data sets, which makes them inflexible. As we discussed, various graphs may require different mechanisms to achieve better receptive field. Using one system cannot fully deal with complex situations. For example, small-range receptive field is more suitable for dense or homophily cases. What if we face a dense but heterophily graph?

To further improve MPNNs' capacity to capture receptive field in different graphs, we propose a novel **MPNN** space for better capturing **D**ata-dependent **R**eceptive **F**ields (MpnnDRF) in this paper. Specifically, we first survey existing solutions to capture receptive fields in different graphs and systemically investigate their capability in various scenarios. Then, we propose several concrete designs to improve existing solutions, including absolute PE/SE-based re-wiring for neighbor mechanism, and relative PE/SE-based attention for weighted receptive field. To fully explore and utilize useful designs in existing works and the proposed ones, we incorporate them into a search space and formulate a search framework, which can search suitable MPNN models to capture the receptive field for the given data. The main contributions are summarized as follows:

Table 1: A summary of notations.

Notation	Definition
$\mathbb{R}^d$	d-dimension real space.
$\lvert\cdot\rvert, \lVert\cdot\rVert$	Length of a set, and l_2-norm.
$G=(V,E,\mathbf{A},\mathbf{X},\mathbf{P})$	A graph with node set V and edge set E, where $\lvert V\rvert = n$ and $\lvert E\rvert = m$.
$\mathbf{A} \in \mathbb{R}^{n\times n}$	The adjacency matrix of G, where $\mathbf{A}[u,v] = 1$ if edge $(u,v) \in E$.
$\mathbf{X} \in \mathbb{R}^{n\times f}$	The matrix of node attributes, and $\mathbf{X}_v \in \mathbb{R}^f$ is the attribute vector of node v.
$\mathbf{P} \in \mathbb{R}^{n\times f_{pe}}$	Matrix of node positional encodings.
$\mathbf{D}=diag(d_1,\cdots,d_n)$	The diagonal degree matrix of $\mathbf{A}$.
$N(v)$	Neighbor set $N(v)=\{u\in V\lvert(u,v)\in E\}$.
$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \in \mathbb{R}^{n\times n}$	Laplacian decomposition of G.
$\mathbf{RW} = \mathbf{A}\mathbf{D}^{-1} \in \mathbb{R}^{n\times n}$	Random walk matrix of G.
$\mathbf{H}^k \in \mathbb{R}^{n\times d}, \mathbf{H}_v^k \in \mathbb{R}^d$	The node representation at layer k.
$\cdot$	Vector dot product.
$\lvert\rvert$	Concatenation operator.
$\sigma(\cdot)$	Non-linear activation function (ReLU).
$tanh(\cdot)$	Hyperbolic tangent activation function.

- We systematically revealed the connection between design dimensions with receptive fields on existing benchmark graph data sets. We develop a novel metric to measure over-squashing and leverage it to investigate existing works with empirical study.
- To better capture receptive fields, we propose two novel designs to improve existing neighbor and weighted receptive field mechanisms, which enable the model to deal with diverse graphs.
- To fully explore existing designs, we implement various design dimensions into one novel MPNN space and bridge the gap between existing AutoMPNNs with the cognition of receptive field.
- Empirical study verifies MpnnDRF is robust to increased receptive field size and less vulnerable to common issues (over-smoothing and over-squashing) in terms of receptive field. On node and graph classification task, MpnnDRF consistently beats baselines across benchmark data sets with diverse properties.

2 RELATED WORK

2.1 MPNNs and Common Issues

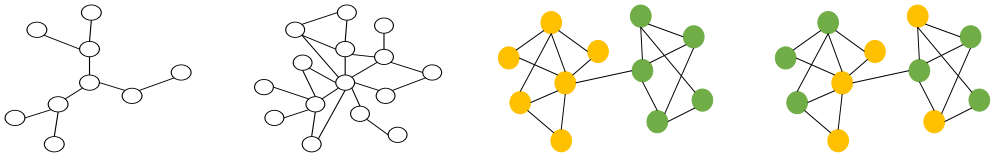
2.1.1 General Message Passing Neural Networks (MPNNs). Following the general message passing schema [23], MPNNs have become the leading approaches on graph representation learning. MPNNs iteratively update node representations $\mathbf{H}$ via message aggregation from immediate (1*hop*) neighbors, and are formalized as:

$$\mathbf{M}_v^{k+1} = agg_k(msg_k(\mathbf{H}_u^k, \mathbf{H}_v^k, e_{uv}^k)|u \in N(v)), \quad (1)$$

$$\mathbf{H}_v^{k+1} = upd_k(comb_k(\mathbf{H}_v^k, \mathbf{M}_v^{k+1})), \quad (2)$$

where $\mathbf{H}_v^0 = \mathbf{X}_v; k \in \{0,1,\ldots,K\}$ is the current MPNN layer/iteration; e_{uv}^k is the weight for edge (u,v); $msg_k(\cdot)$ is the message function that computes messages; $agg_k(\cdot)$ is the aggregation function that gathers messages from 1*hop* neighbors $N(v) = \{u \in V|(u,v) \in E\}$ to induce immediate representation $\mathbf{M}_v^{k+1}$; $comb_k(\cdot)$ function combines neighbors' information with node itself, which is then fed into update function $upd_k(\cdot)$ (general non-linear) to obtain $\mathbf{H}_v^{k+1}$. After each iteration, receptive field of node v is enlarged as:

$$\mathcal{N}_{k+1}(v) = \mathcal{N}_k(v) \cup \{u \in V|\mathbf{A}[w,u] = 1 \cap w \in \mathcal{N}_k(v)\}. \quad (3)$$



(a) Sparse. (b) Dense. (c) Homophily. (d) Heterophily.

Figure 2: Graphs with different densities. Graphs with homophily/heterophily patterns.

After K-th iterations, MPNNs yield learned representations $\mathbf{H}_v = \mathbf{H}_v^K$. For node-set learning tasks, a permutation-invariant readout function $read(\cdot)$ is further required to induce high-order representations. E.g., for graph classification, $read(\cdot)$ maps $\mathbb{R}^{n\times d} \rightarrow \mathbb{R}^d$.

Different MPNNs mainly differ from each other in terms of main components in Eq. (1) and (2). E.g., GCN [30] instantiates Eq. (1) and (2) as follows, with $\mathbf{W}_k$ as trainable transformation matrix.

$$\mathbf{M}_v^{k+1} = \sum\nolimits_{u\in\tilde{N}(v)} \mathbf{H}_u^k / \sqrt{\tilde{d}_u \tilde{d}_v}, \quad \mathbf{H}_v^{k+1} = \sigma(\mathbf{W}_k \mathbf{M}_v^{k+1}). \quad (4)$$

2.1.2 Common issues of general MPNNs in terms of receptive field. As discussed in Sec. 1, general MPNNs may be incapable of handling variant receptive fields required in different graph scenarios. For example, dense/homophilic graphs may be vulnerable to over-smoothing, where node representations converge to stationary distribution and become indistinguishable. Heterophilic graphs may encounter another prominent issue, i.e., over-squashing, referring to MPNNs' inability to propagate distant messages due to the bottleneck originating from original graph topology [2, 62]. Although both over-smoothing and -squashing are arguably caused by the failure of given node's learned representation being affected by other key nodes and hinder MPNN results, they are generally more relevant to short-range and long-range (range means interaction hop between nodes) graph scenarios respectively [2]).

Decoupled MPNNs (DMPNNs). Early MPNNs follow coupled way to integrate non-linear feature transformation $upd(\cdot)$ with topology aggregation $agg(\cdot)$ (Eq. (1) and (2)). However, recent works challenge the necessity of such coupled designs from the view of model performance (e.g., over-smoothing) and efficiency, especially for deeper models. Thus, they propose Decoupled MPNNs (DMPNNs) [8, 31, 42, 53, 70, 79] as solutions. As a representative, APPNP [31] is formulated as Eq. (5), where $f_\theta(\cdot)$ is parametric update function and $\beta \in [0, 1]$ is teleport probability to preserve locality.

$$\mathbf{H}_v^0 = f_\theta(\mathbf{X}_v), \quad \mathbf{H}_v^{k+1} = \beta\mathbf{H}_v^0 + (1-\beta)\sum\nolimits_{u\in\tilde{N}(v)} \mathbf{H}_u^k / \sqrt{\tilde{d}_u \tilde{d}_v}. \quad (5)$$

Non-local MPNNs. To alleviate potential issues (e.g., over-squashing) and better capture receptive fields, non-local mechanisms have been developed recently, e.g., high-order MPNNs [1, 8, 11, 53, 70, 83], geometric MPNNs [50], fully-connected MPNNs [2], and Graph Transformers [16, 34, 45].

2.2 Automated MPNNs

Despite success, general (manual) MPNNs are limited to fixed message passing mechanism for all graphs and may lead to data-aware issue, i.e., graph with different properties may require different MPNN mechanisms for better graph representation learning.

To tackle this, Automated Message Passing Neural Networks (AutoMPNNs) [22, 67, 68, 79] have been recently developed. Generally, AutoMPNNs focus on searching key functions in Eq. (1) and (2) (e.g., $agg_k(\cdot)$) to build optimal MPNN structure for given graph data. Specifically, AutoMPNNs construct operator search space $\mathcal{O}$ for these functions (design dimensions) and propose candidate-sets (design choices) accordingly. For simple illustration (we refer readers to [81] for comprehensive survey), we list some common design choices: $\mathcal{O}_N = \{first\_order\}, \mathcal{O}_e = \{non\_norm, sys\_norm, rw\_norm\}$, $\mathcal{O}_{msg} = \{e_{uv}\mathbf{H}_u\}, \mathcal{O}_{agg} = \{sum, mean, max, min\}, \mathcal{O}_{comb} = \{sum, concat\}$, and $\mathcal{O}_{upd} = \{ReLU(\mathbf{W}), PReLU(\mathbf{W})\}$. Then, various search algorithms (e.g., differentiable [41, 71]) can be used to search the optimal MPNN from defined $\mathcal{O}$. However, existing AutoMPNNs neglect important design dimensions for receptive field, and thus may be incapable to handle diverse receptive fields required in different graphs and vulnerable to common issues, e.g., over-smooth and -squashing.

3 METHODOLOGY

In Sec. 3.1, we first introduce our proposed novel quantitative metric to access over-squashing. In Sec. 3.2, we present several motivational experiments to support the arguments in Sec. 1. Then, motivated by those experiments, we propose a comprehensive space in Sec. 3.3 to cover both existing design dimensions and some designs improved by us. Finally, in Sec. 3.4, we formulate a search problem and propose to search a suitable MPNN model for the given graph.

3.1 The Proposed Metric for Over-squashing

As mentioned in Sec. 2.1.2, over-smoothing and -squashing are common issues of general MPNNs in terms of receptive field. While over-smoothing is more extensively explored with various quantitative metrics being developed, e.g., *MAD* [7] (see Appx. A.1), over-squashing is less studied with few metrics on hand. [62] proposes Jacobian $I(v, u)$ (see Appx. A.1) to access over-squashing effect for individual nodes. Unfortunately, unlike *MAD* as a graph-level (global) metric, $I(v, u)$ is intrinsically a node-level (local) one, making it difficult to quantify the overall over-squashing degree of the entire graph given certain MPNN designs.

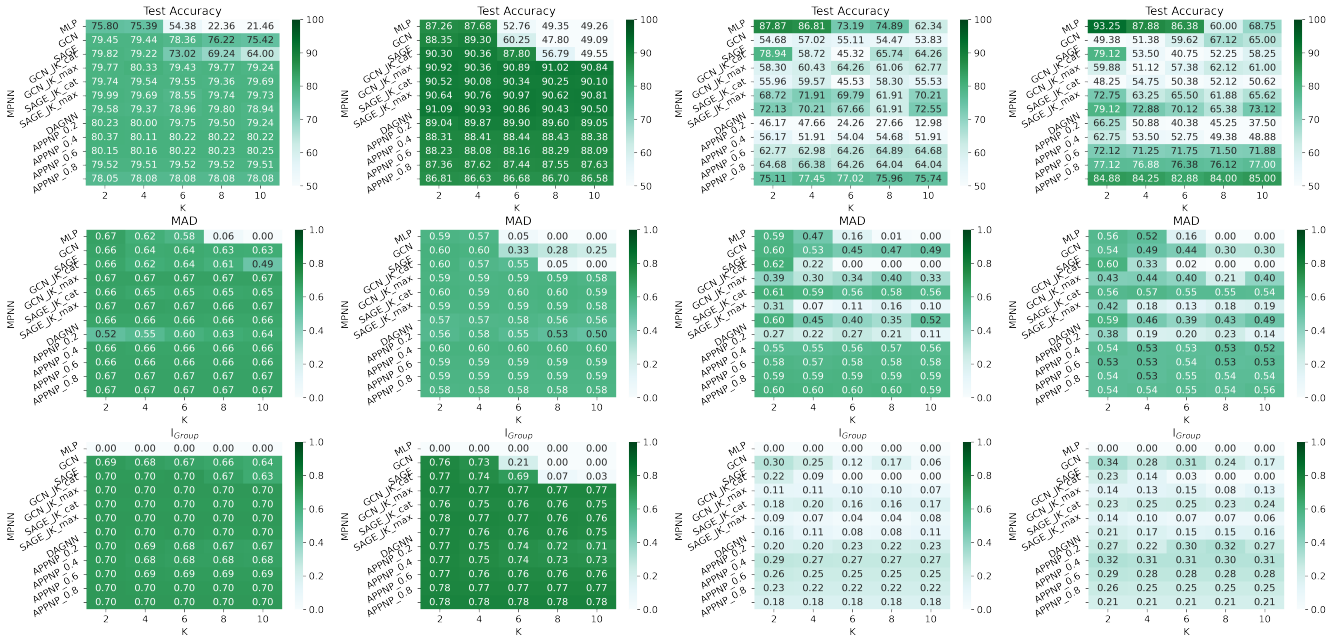
To bridge this gap, we develop a novel graph-level quantitative metric $I_{Group} \in [0, 1]$ for graph G built on existing $I(v, u)$:

$$I_{Group} = \frac{\sum_{u,v\in V \wedge y_u=y_v} I_G[v, u]}{\sum_{u,v\in V} I_G[u, v]} = \frac{\sum_{v\in V}\sum_{u\in V\wedge y_u=y_v} I_v[u]}{n}, \quad (6)$$

where $I_G = [I_{v_1}, I_{v_2}, \ldots, I_{v_n}] \in \mathbb{R}^{n\times n}$ is the matrix that collects influence distributions $I_v \in \mathbb{R}^n$ for $v \in V$, and $I_v[u] = I(v, u)/\sum_{z\in V} I(v, z)$. I_{Group} measures the average proportion of influence from intra-class messages against all (intra-class as well as inter-class) messages based on the induced receptive field of specific MPNN design.

3.2 Empirical Observation of Existing MPNNs

3.2.1 Experimental settings. As discussed in Sec. 1 and Sec. 2.1, MPNNs rely on the receptive field to aggregate information for



(a) Citeseer (b) Pubmed (c) Cornell (d) Wisconsin

Figure 3: Empirical results (test accuracy, quantitative MAD and I_{Group}) of several advanced MPNNs.

node representation learning. There are two main differences in how to capture receptive fields in existing works: 1) how receptive field expands at each iteration/layer; 2) how many expansion iterations K are required. The majority of existing works rely on $1hop$ mechanism to expand receptive field (i.e., iteratively aggregate $1hop$ neighbors based on original graph connectivity $\mathbf{A}$) since they assume $\mathbf{A}$ is highly informative for representation learning. Early methods usually set small K since large K would lead to oversized receptive field, raising risks of performance collapse due to over-smoothing. Some later improvements (e.g., DMPNN; MPNN with adaptive layers) propose new MPNN designs and try larger receptive field for more information and higher performances.

To explore some classic MPNNs, we utilize common benchmarks of node classification for evaluation: Citeseer, PubMed, Cornell, and Wisconsin (statistics in Tab. 7). Citeseer and PubMed are homophily graphs while the rest are heterophily ones ($homo.ratio = \frac{|(u,v):(u,v)\in E \wedge y_u = y_v|}{|E|} \in [0,1]$, y_v is label for node v). Adopted $1hop$ methods (except for MLP that are $0hop$) are as follows, which are different in terms of: 1) coupled/decoupled for aggregation and transformation, 2) mechanism ($inter\_agg$) to fuse multi-scale information from different range of receptive fields (iterations/layers).

- MLP: no aggregation (only transformation), no $inter\_agg$;
- GCN, SAGE: coupled, no $inter\_agg$;
- GCN_JK_cat, SAGE_JK_cat: coupled, non-adaptive $inter\_agg$;
- GCN_JK_max, SAGE_JK_max: coupled, node-adaptive $inter\_agg$; coupled MPNN and layer-adaptive;
- DAGNN: decoupled, node-adaptive $inter\_agg$;
- APPNP with different teleport probability $\beta \in \{0.2, 0.4, 0.6, 0.8\}$ (see Eq. (5)): decoupled, non-adaptive $inter\_agg$.

3.2.2 Observations and analysis. Empirical results under variant K (2-10 as in x-axis) are shown in Fig. 3. Note that K means maximum allowed layers for adaptive works, and K is fixed in other works.

From Fig. 3(a) and 3(b), we first observe that performance collapse occurs for MLP, GCN and SAGE as layer increases. For GCN and SAGE, this is because the increased layers will lead to a larger receptive field, which will make the node representations difficult to distinguish (small MAD and over-smoothing). However, for ($0hop$) MLP that only contains the given node itself within the receptive field no matter K settings, it just works as feature transformation and the performance collapse cannot be attributed to over-smoothing. Instead, it may be caused by the vanishing gradient problem or training difficulty of deeper NNs. This indicates that deep layers of feature transformation may also deteriorate performances. Despite techniques in other more advanced MPNNs (e.g., decoupled schema and adaptive layers) are different, their performance is robust to different K settings (2-10). Besides, we obtain a high Pearson correlation coefficient [10] (0.59-0.67) between MAD and test performance for them. This may indicate that their efforts in terms of receptive field designs (e.g., decoupled schema and adaptive layers) empower them to alleviate over-smoothing, thereby achieve superior and robust performances when dealing with homophily graphs.

From Fig. 3(c) and 3(d), however, we observe significantly different patterns. Specifically, we notice that these advanced MPNNs perform less robust, and their advantages over GCN and SAGE become less significant compared with Fig. 3(a) and 3(b). This may indicate their inability on heterophily graphs. To seek for potential explanations, we take a further look at values of quantitative MAD and I_{Group}. We find that on Cornell, and Wisconsin, adopted (1hop) MPNNs lead to a wide spectrum of MAD (0.10-0.60) but

consistently low I_{Group} (0.16-0.21). This may indicate their inferior results are more originated from over-squashing rather than over-smoothing. We further conjecture that the inadequate $1hop$ mechanism may be the root cause. One supporting finding for this hypothesis is that, the simplest ($0hop$) MLP achieves surprisingly better and robust results than other ($1hop$) counterparts. To further verify, we then set simple variants in terms of neighbor mechanism: classic GCN aggregation coped with $1hop$, $2hop$, and knn neighbor mechanism ($1hop$ setting recovers the vanilla GCN) that are popular in some latest works [29, 83]. From Fig 4 (Appx.), we now obtain a high Pearson correlation coefficient (0.54-0.77) between I_{Group} and test accuracy. Besides, $2hop$ and knn outperform classic $1hop$ probably due to their capacity to alleviate over-squashing (large I_{Group}). However, such superiority diminishes as K increases (2->10), indicating the occurrence of over-smoothing. Coping with simple $inter\_agg$ helps retain their advantages even with larger K. Additionally, even within one model, optimal settings may also differ across datasets. For instance, APPNP with $\beta \in \{0.2, 0.4\}$ are better in Fig. 3(a) and 3(b), while APPNP with $\beta \in \{0.6, 0.8\}$ are more suitable in Fig. 3(c) and 3(d).

To summarize, we verify our key arguments in Sec. 1 via empirical observations. We thus conclude that a more comprehensive, powerful, and flexible framework for receptive field designs is much desired so as to deal with complex graph scenarios.

3.3 The Space of Design Dimensions for Capturing Receptive Field

In Sec. 3.2, we observe that various models perform differently on different graph sets. We assume that this is caused by different specific designs of the model. Therefore, in this subsection, we first study different model designs in MPNNs for capturing the receptive field. Then, we propose specific designs to improve them. Finally, we summarize the proposed designs and form a search space.

Similar to Eq. (1) and (2), we first summarize the MPNN framework finally adopted in this paper as Eq. (7), (8), and (9). Note that classic MPNNs mainly focus on how to design specific operators within one layer, i.e., intra-layer designs. But some recent study [42] show that designing connectivity between different layers can effectively deepen the MPNNs layers and alleviate over-smoothing. Therefore, we also include inter-layer designs as follows:

$$Pre\text{-}MPNN: \quad \mathbf{H}_v^0 = f_\theta(\mathbf{X}_v), \tag{7}$$

$$Intra\text{-}layer: \begin{cases} \mathbf{M}_v^{k+1} = intra\_agg_k(e_{uv}^k \mathbf{H}_u^k | u \in \bar{N}_k(v)), \\ \mathbf{H}_v^{k+1} = update_k(comb_k(\mathbf{H}_v^k, \mathbf{M}_v^{k+1})), \end{cases} \tag{8}$$

$$Inter\text{-}layer: \quad \mathbf{H}_v = inter\_agg(\mathbf{H}_v^0, \ldots, \mathbf{H}_v^K). \tag{9}$$

In this paper, we identify that the choices of $\bar{N}_k(v)$, and e_{uv}^k are important to MPNNs to capture the data-specific properties for the receptive field. Therefore, we will focus on explaining existing designs of them and how to improve them. As for other design dimensions, we follow the common space as introduced in Sec. 2.2, such as $O_{intra\_agg} \in \{sum, mean, max, min\}$ and $O_{comb} \in \{sum, concat\}$.

3.3.1 Transformation. Motivated by Sec. 3.2, we found that the recently developed decoupled MPNNs (e.g., DAGNN and APPNP) achieve good performance on a wide range of graph data sets with different properties. Thus, we include it in Eq. (7). We follow the common practice in DMPNNs to instantiate $f_\theta(\cdot)$ as the MLP model. More specifically, based on empirical observations in Fig. **??** and Sec. 3.2 that deep layers of feature transformation may deteriorate performances, we set the layers of MLP as 2.

Table 2: Receptive field size under two-hop setting: naive (without pruning) method and MpnnDRF (with pruning).

Property	Acto.	Corn.	Texa.	Wisc.	Cite.	PubM.	DBLP	CS
Naive	1279966	4949	5973	8539	22410	562892	738306	985910
MpnnDRF	62806	221	287	557	4737	86184	164724	228780
ratio	4.9%	4.5%	4.8%	6.5%	21.1%	15.3%	22.3%	23.2%

3.3.2 Neighbors mechanism $\bar{N}_k(v)$. As discussed in Sec. 1 and Sec. 2.1, conventional MPNNs directly aggregate messages from the first-order ($1hop$) neighbors $N(v) = \{u \in V | \mathbf{A}[u, v] = 1\}$ (Eq. (1)).

High-order neighbors. To exploit high-order ($lhop$) neighbors, some works [1, 8, 11, 29, 53, 70, 82, 83] go beyond the first-order restriction to better capture long-range dependencies:

$$\bar{N}_k(v) = \{u \in V | \mathbf{A}^l[u, v] = 1\}. \tag{10}$$

However, this may result in an exponentially large neighborhood size and increase computational cost. Thus, we leverage pruning strategy as in Eq. (11) and constraint $l \leq 2$ to alleviate it. As shown in Tab. 2, MpnnDRF's pruning method significantly reduces the exponential issue when leveraging high-order neighbors.

$$\bar{N}_k(v) = \{u \in V | \mathbf{A}^l[u, v] \geq l\}. \tag{11}$$

Re-wiring neighbors. The first/high-order neighbor mechanisms still follow the original graph connectivity $\mathbf{A}$ to aggregate neighbors. Instead, some works propose to re-wire $\mathbf{A}$ into $\hat{\mathbf{A}}$ to denoise the graphs (i.e., revising $\mathbf{A}[u, v] = 1$ to 0) and ease the message flow between distant nodes [32, 45] (revising $\mathbf{A}[u, v] = 0$ to 1 where u and v are distant in the original graph).

Because changing the original connectivity may greatly distort the information of the graph, different methods adopt different strategies to modify the connectivity of the graph, including diffusion-based (personalized PageRank [48] and heat-kernel [33]) and attribute-based re-wiring methods. The main idea of their methods is to learn a similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ for any node pairs in the graph. In diffusion-based re-wiring methods, $\mathbf{S}_{gdc}$ is defined as:

$$\mathbf{S}_{gdc} = \sum_{l=0}^{\infty} w_l \mathbf{A}^l, \tag{12}$$

where w_l defines coefficient based on PPR gdc_ppr and heat-kernel gdc_heat. β is teleport probability, and t is diffusion time.

$$w_l = \begin{cases} \beta(1-\beta)^l, & gdc\_ppr, \\ e^{-t}\frac{t^l}{l!}, & gdc\_heat, \end{cases} \tag{13}$$

Different from above methods relying on graph connectivity, attribute-based re-wiring methods argue that the graph connectivity may exhibit limited informativeness [29, 66]. Thus, they propose to reconstruct the graph based on similarity of node attributes:

$$\mathbf{S}_{attr}[u, v] = sim(\mathbf{X}_u, \mathbf{X}_v), \tag{14}$$

where Cosine similarity can be employed as the similarity matrix $sim(\cdot)$. However, the graph may be too dense if we derive the graph connectivity based on $\mathbf{S}_{gdc}$ and $\mathbf{S}_{attr}$. Thus, kNN [29, 32, 66] is further applied on $\mathbf{S}_{gdc}$ and $\mathbf{S}_{attr}$ to build the graph connectivity. Taking $\mathbf{S}_{gdc}$ as an example, the derivation of connectivity and

Table 3: Search space of MpnnDRF: design dimensions and candidate-sets.

Type	Design dimension		Candidate-set O
Pre-MPNN	Transformation	f_θ	$O_{f_\theta} = \{MLP\}$
Intra-layer	Neighbor mechanism	$\tilde{N}_k(v)$	$O_N = \{first\_order, high\_order, attr\_rewire, diffuse\_rewire, PE/SE\_rewire\}$
	Edge weight	$\mathbf{e}_{uv}^k$	$O_e = \{non\_norm, sys\_norm, rw\_norm, self\_gating, rel\_lepe, rel\_rwpe\}$
	Intra aggregation	$intra\_agg_k$	$O_{intra\_agg} = \{sum, mean, max, min\}$
	Combination	$comb_k$	$O_{comb} = \{sum, concat\}$
Inter-layer	Inter aggregation	$inter\_agg$	$O_{inter\_agg} = \{concat, last, mean, decay, gpr, lstm\_att, gating\}$

neighbors can be formed as:

$$\mathbf{A}_{gdc} = k\text{NN}(\mathbf{S}_{gdc}), \tilde{N}_k(v) = \{u \in V | \mathbf{A}_{gdc}[u, v] = 1\}, \quad (15)$$

where $\mathbf{S}_{gdc}$ (dense) $\rightarrow \mathbf{A}_{gdc}$ (sparse).

However, attribute-based re-wiring methods cannot work on non-attributed graphs, which is common among popular graph classification data. And diffusion-based re-wiring methods may be inappropriate when dealing with heterophilic graphs [62].

Re-wiring based on positional/structural encoding.

As discussed, existing high-order neighbor mechanism may suffer from exponentially large neighborhood size, and existing re-wiring mechanism may be incapable when graphs exhibit certain properties (e.g., non-attributed). To improve, we propose positional/structural encoding (PE/SE)-based re-wiring methods. PE/SE has been shown powerful in defining MPNN directions that are capable to counteract over-smoothing and -squashing, shorten diffusion distance and improve MPNN efficiency [3]. Recently, PE/SE has been adopted in Graph Transformers (GTs) [46] and MPNNs [3, 65], which generally works as raw attributes or edge weights (attentions). However, the way that existing works incorporate PE/SE as soft inductive bias (edge weight) in GTs or 1hop MPNNs may be suboptimal, since fully-connected GTs may be costly due to oversized receptive field, and sparse 1hop MPNNs may be less flexible due to the fixed receptive field. Inspired by existing designs, we reason that leveraging PE/SE to re-wire neighbors under the sparse MPNN schema may be promising to better capture receptive field:

$$\mathbf{S}_{pe}[u, v] = sim(\mathbf{P}_u, \mathbf{P}_v). \quad (16)$$

Specifically, we leverage Laplacian eigenvector as PE since it allows efficient message passing [3], and random-work matrix as SE [18, 39] since it demonstrates powerful to model substructures in graphs:

$$\mathbf{P}_v = \begin{cases} [\mathbf{U}_{v1}, \mathbf{U}_{v2}, \dots] \in \mathbb{R}^{d_{pe}}, & lepe, \\ [\mathbf{RW}^1_{vv}, \mathbf{RW}^2_{vv}, \dots] \in \mathbb{R}^{d_{se}}, & rwse. \end{cases} \quad (17)$$

where $\mathbf{U}$ is the eigenvector matrix, and $\mathbf{RW} = \mathbf{AD}^{-1}$. Then, analogously as in Eq. (15), kNN is further applied to the similarity matrix $\mathbf{S}_{pe}$ to build the sparse graph connectivity $\mathbf{A}_{pe}$.

3.3.3 Edge Weight e_{uv}^k. MPNNs initially do not model the edge weights as they are mainly designed for homogeneous graph. But recent studies show that edge weight can control the message passed from $\mathbf{H}_u^k$ to $\mathbf{M}_v^{k+1}$ (see Eq. (7)), which can help control the magnitude of messages. Here we first introduce two common designs, non-parametric and attentive modelings.

Non-parametric weight. Utilizing node degree to normalize the weight e_{uv}^k is widely adopted in mainstream MPNNs [24, 30, 72], e.g., GIN [72] directly assigns the constant weight for all messages; GCN [30] utilizes systematic normalization; SAGE [24] normalizes based on the random-walk process. We summarize such approaches as Eq. (18), where $\tilde{d}_u$ is the degree for node u (with self-loop).

$$e_{uv}^k = f_e(\tilde{d}_u, \tilde{d}_v) = \begin{cases} 1, & non\_norm, \\ 1/\sqrt{\tilde{d}_u \tilde{d}_v}, & sys\_norm, \\ 1/\tilde{d}_u, & rw\_norm. \end{cases} \quad (18)$$

Attentive weight. Recent works [2] observe that MPNNs with attention e.g., GAT [63] with self-attention, may be less susceptible to over-squashing. Thus, we model the weight e_{uv}^k with representations of source and target nodes (i.e., $\mathbf{H}_u$ and $\mathbf{H}_v$) as follows:

$$e_{uv}^k = f_e(\mathbf{H}_u, \mathbf{H}_v) = tanh(\mathbf{g}^T \cdot [\mathbf{H}_u^k || \mathbf{H}_v^k]), \quad self\_gating \quad (19)$$

where $\mathbf{g} \in \mathbb{R}^{2d}$ is the linear projection vector that to be learned; $tanh(\cdot)$ induces real valued e_{uv}^k ranging between $[-1, 1]$. Compared against GAT which generally impose the non-negative constraint on e_{uv}^k, we allow the magnitude as well as sign of message (e_{uv}^k in Eq. (19)) to be more adaptive, which may be more flexible to handle high-frequency graph signals [3, 4].

PE/SE weight. Non-parametric weight that simply depends on degree information may suffer from limited capacity, and attentive weight cannot handle graphs when raw attributes are not available or uninformative. Hence, analogously with proposed PE/SE-based re-wiring, we further provide another PE/SE-based method for better capture receptive field, i.e., we propose to leverage relative PE/SE to control the edge weights, which may be regarded as soft version of proposed PE/SE-based re-wiring methods:

$$e_{uv}^k = f_e(\mathbf{P}_u, \mathbf{P}_v) = \phi(\|\mathbf{P}_u - \mathbf{P}_v\|) \quad (20)$$

$$= \begin{cases} \phi(\|\mathbf{P}_u^{lepe} - \mathbf{P}_v^{lepe}\|), & rel\_lepe, \\ \phi(\|\mathbf{P}_u^{rwse} - \mathbf{P}_v^{rwse}\|), & rel\_rwse, \end{cases} \quad (21)$$

where the relative positional encoding $\|\mathbf{P}_u - \mathbf{P}_v\|$ measures the relative distance between $\mathbf{P}_u$ and $\mathbf{P}_u$ and is derived via the l_2-norm $\|\cdot\|$ ($\mathbb{R}^d \rightarrow \mathbb{R}$); ϕ is the neural network that transforms relative PE/SE to induce scalar weight ($\mathbb{R} \rightarrow \mathbb{R}$) for the incoming message.

3.3.4 Inter-layer Receptive Field Fusion. Recent works observe that fusing (aggregating) multi-scale information from different range of receptive fields (layers) may be beneficial towards richer representation learning. Hence, MpnnDRF further covers such inter-layer design dimension $inter\_agg$ in Eq. (9) to support the inter-layer message passing. Motivated by the recent works [8, 31, 36, 36, 48, 53, 73, 82], the operator space of $inter\_agg$ in this paper is denoted to $inter\_agg \in \{concat, last, mean, decay, gpr, lstm\_att, gating\}$. Next we roughly categorize them into three types from the perspective of how they aggregate information and whether the weight of aggregate information is adaptive.

Layer-wise, non-adaptive. Layer-wise methods mainly aggregate the representations of different layers to learn the representation, like $inter\_agg(\cdot) = concat(\mathbf{H}_v^0, \dots, \mathbf{H}_v^K) = [\mathbf{H}^0 || \cdots || \mathbf{H}^K]$.

And $\{last, mean, decay\}$ methods can be formed as weighted sum $\mathbf{H} = \sum_{k=0}^{K} w_k \mathbf{H}^k$, where w_k is defined as:

$$w_k = \begin{cases} \delta_{kK}, & last, \\ 1/(K+1), & mean, \\ \beta(1-\beta)^k, & decay. \end{cases} \tag{22}$$

In Eq. (22), *last* [30, 70] completely ignores the inter-layer NMP and directly takes representations from last layer as final output, i.e., $w_k = \delta_{kK}$, where $\delta_{kK} = 1$ if $k = K$ otherwise 0; *mean* [82] assigns equal weights for each layer with $w_k = 1/(K+1)$; *decay* [8, 31] follows the personalized PageRank (PPR) [48], which penalizes the importance of deeper layers to preserve locality as $w_k = \beta(1-\beta)^k$, where $\beta \in [0, 1]$ is the teleport probability. Since the weights of combining different layers (w_k) are non-parametric, this type of operators belong to non-adaptive ones.

Layer-wise, adaptive. In Eq. (22), w_k is preset based on some prior knowledge and restricted to be non-negative. Despite the simplicity, such methods may lead to limited flexibility. As demonstrated in [9], the dissolution of non-negative constrain for w_k brings further benefit to better tackle the low-frequency/high-frequency aspects of graph signals simultaneously. Inspired by it, we further allow the weights between layers to be learnable (denoted as γ_k) as: $\mathbf{H} = \sum_{k=0}^{K} \gamma_k \mathbf{H}^k$, where γ_k is trained end-to-end and can be both positive/negative.

Node-wise, adaptive. Recent works [73, 78, 80] observe that the expansion speed of RF among different nodes may be inconsistent when graph exhibits significantly heterogeneous local patterns (e.g., sparsity). Thus, fixed *inter_agg* for all nodes may lead to the oversized RF (with excessive irrelevant noises) for some nodes, and the undersized RF (with insufficient messages) for others. To further boost the flexibility of *inter_agg*, node-wise adaptive candidates $\{lstm\_att, gating\}$ are integrated. Similar to layer-wise adaptive methods, $\mathbf{H}_v$ is formulated as $\mathbf{H}_v = \sum_{k=0}^{K} \gamma_v^k \mathbf{H}_v^k$, where $\{\gamma_v^0, \ldots, \gamma_v^K\}$ are personalized coefficients for node v. Generally, $\gamma_v^k \neq \gamma_u^k$, $\forall u, v \in V$. More specifically:

$$\gamma_v^k = \begin{cases} softmax(\{\mathbf{a}^T \cdot [\mathbf{F}_v^k || \mathbf{B}_v^k]\}_{k=0}^{K}), & lstm\_att, \\ tanh(\mathbf{c}^T \cdot \mathbf{H}_v^k), & gating. \end{cases} \tag{23}$$

As shown in Eq. (23), $lstm\_att$ follows [73] to yield attention scores γ_v^k, where $\mathbf{F}_v^k/\mathbf{B}_v^k$ denotes forward/backward representation output from a bidirectional LSTM model [26], $\mathbf{a} \in \mathbb{R}^{2d}$ is the linear projection vector, and $softmax$ is the softmax function. *gating* follows the gating mechanism to obtain γ_v^k via the learnable calibration vector $\mathbf{c} \in \mathbb{R}^d$, which is global and uniform for all nodes. Note that different from previous approaches which typically restrict γ_v^k to be non-negative, here we leverage the $tanh$ function to enable both signs of coefficients for better flexibility.

In summary, we propose two novel design dimensions, including PE/SE-based re-wiring for $\bar{N}_k(v)$, and PE/SE-based weight for e_{uv}^k, to improve MPNNs' capability in handling receptive field. For other inter-layer and intra-layer design candidates, we incorporate powerful designs in existing works. As shown in Tab. 3, we summarized the adopted designs in our search space $\mathcal{O}$. Overall, a MPNN model $\mathcal{M}$ can be combined by different combinations from our space $\mathcal{O}$.

3.4 Search Framework

As observed in Sec. 3.2, different graph properties can influence the required receptive fields in MPNNs. But most of existing solutions aim to propose a common solution for different graph data sets, which makes them inflexible to capture complex receptive fields. For example, homophilic graphs may benefit from diffusion-based graph re-wiring, and non-attributed graphs may require PE/SE-based graph re-wiring. Even in one model, when facing different data, the model design requires a large change to deal with the needs of the receptive field. Motivated by such observations, we propose to incorporate the discussed designs and existing designs in Sec. 3.3 into a big operator space (Tab. 3). Formally, the objective of the proposed MpnnDRF can be formulated as:

$$\boldsymbol{\alpha}^*, \boldsymbol{\omega}^* = \arg\max_{\boldsymbol{\alpha}, \boldsymbol{\omega}} \mathbb{E}_{\mathcal{M} \sim \pi_{\boldsymbol{\alpha}}} f(\mathcal{M}, \boldsymbol{\omega}; D), \tag{24}$$

where π is the AutoGNN-controller parameterized by architecture parameters $\boldsymbol{\alpha}$, and is used to control the search of optimal MPNN architecture $\mathcal{M}$ from the defined search space $\mathcal{O}$ (see Tab. 3); $f(\mathcal{M}, \boldsymbol{\omega}; D)$ evaluates the performance of $\mathcal{M}$ with network weights $\boldsymbol{\omega}$ on graph data D with metric $f(\cdot)$; $\mathbb{E}$ is the expectation function.

Searching for optimal$\mathcal{M}$ is equivalent to finding the best instantiation of MPNN process shown in Eq. (7) and (9), which can be further decomposed as selecting the most suitable candidate for each design dimension. Given a specific design dimension, let $\mathcal{O}$ be its candidate-set with size $|\mathcal{O}| = N$, and p_α denotes the architecture distribution to be optimized, which is parameterized by $\alpha \in \mathbb{R}^N$ with each element $\alpha_o \in \mathbb{R}$ recording the importance of candidate o, $\forall o \in \mathcal{O}$. Then given the immediate representation $\mathbf{z}$ as input for this dimension, the forward-propagation yields the output as:

$$\bar{o}(\mathbf{z}) = \sum_{o \in \mathcal{O}} \mathcal{M}_o \cdot o(\mathbf{z}), \tag{25}$$

where $\mathcal{M}_o \in \{0, 1\}$ and we let $\sum_{o \in \mathcal{O}} \mathcal{M}_o = 1$. The candidate selection process $\{\mathcal{M}_o\}_{o \in \mathcal{O}}$ is controlled by p_α and intrinsically discrete, which makes it unfeasible to directly back-propagate the gradient of α for optimization via Eq. (25). To tackle this, the concrete distribution [27, 44] is leveraged for unbiased approximation [71], which adopts the re-parameterization trick as in Eq. (26) to relax the discrete candidate selection (sampling) process to be continuous and thus differentiable, where τ is the temperature for $softmax$ function, and $U_o \sim Uniform(0, 1)$ denotes the uniform sampling.

$$\mathcal{M}_o = softmax((\log \alpha_o - \log(-\log(U_o)))/\tau). \tag{26}$$

Empirically, we minimize the cross entropy loss function $\mathcal{L}(\cdot)$ as in Eq. (27) to solve our optimization objective (Eq. 24), where C is the number of classes of graph dataset D, $\mathbf{Z}_i \in \mathbb{R}^C$ is the output prediction for data sample (node or graph) i based on $\mathcal{M}$, and $y_i \in \mathbb{R}$ is the label for data sample i.

$$\mathcal{L}(\mathcal{M}, \omega; D) = \frac{1}{|D|} \sum_{i=1}^{|D|} -log(\frac{\exp(\mathbf{Z}_i[y_i])}{\sum_{j=0}^{C-1} \exp(\mathbf{Z}_i[j])}). \tag{27}$$

4 EXPERIMENTS

4.1 Experimental Settings

Task and data sets. For node classification task, we adopt 8 benchmark data sets with edge homophily ratio covering the whole spectrum from 0 to 1. Dataset statistics and splits are provided in Tab.

Table 4: Model performance (accuracy with standard derivation) on the node classification task.

Type	Model	Actor	Cornell	Texas	Wisconsin	Citeseer	PubMed	DBLP	CS
MLP	MLP	38.58 (0.25)	91.36 (0.70)	92.26 (0.71)	92.26 (3.04)	73.82 (1.00)	86.43 (0.13)	77.57 (0.69)	89.51 (0.46)
Manual MPNNs	GCN	30.59 (0.23)	66.72 (1.37)	75.16 (0.96)	69.56 (5.42)	79.21 (1.22)	86.97 (0.12)	85.91 (0.56)	91.93 (0.37)
	GAT	35.98 (0.23)	76.00 (1.01)	78.87 (0.86)	73.08 (5.10)	80.56 (0.31)	86.64 (0.11)	88.18 (0.88)	90.89 (0.47)
	SAGE	36.37 (0.21)	71.41 (1.24)	79.03 (1.20)	91.58 (3.32)	78.24 (0.30)	86.85 (0.11)	85.96 (0.59)	92.48 (0.21)
	GBK-GNN	38.97 (0.97)	74.27 (2.18)	81.08 (4.88)	84.21 (4.33)	79.18 (0.96)	89.11 (0.23)	85.07 (0.39)	94.15 (0.31)
	SGC	29.39 (0.20)	47.80 (1.50)	55.18 (1.17)	49.94 (6.28)	76.23 (0.29)	83.52 (0.10)	85.74 (0.45)	90.94 (0.47)
	ChebNet	38.02 (0.23)	85.33 (1.04)	86.08 (0.96)	91.31 (3.40)	78.66 (0.26)	88.20 (0.09)	85.15 (0.58)	92.63 (0.26)
	H_2GCN	35.86 (1.03)	82.16 (4.80)	84.86 (6.77)	86.67 (4.69)	76.72 (1.50)	88.50 (0.64)	86.12 (3.14)	95.35 (0.32)
	WRGAT	36.53 (0.77)	81.62 (3.90)	84.86 (6.77)	86.98 (3.78)	76.81 (1.89)	88.52 (0.92)	85.37 (0.62)	94.74 (0.08)
	APPNP	38.86 (0.24)	91.80 (0.63)	91.18 (0.70)	92.14 (2.91)	80.74 (0.94)	89.15 (0.13)	86.90 (0.67)	91.96 (0.27)
	GPRGNN	39.30 (0.27)	91.36 (0.70)	92.92 (0.61)	92.20 (3.42)	80.01 (0.28)	89.18 (0.15)	86.64 (1.08)	91.86 (0.41)
Auto MPNNs	AutoGEL	42.79 (0.95)	90.62 (2.91)	90.44 (2.81)	91.62 (2.67)	80.71 (0.96)	90.43 (0.66)	91.34 (1.32)	94.82 (0.37)
	MpnnDRF	42.97 (0.81)	**92.99 (2.62)**	**95.08 (1.20)**	**94.83 (1.89)**	**81.08 (0.74)**	**90.47 (0.44)**	**92.69 (1.47)**	**96.06 (0.55)**

7 (Appx. A.2). Among them, Actor [61] is actor co-occurrence network; Cornell, Texas, and Wisconsin [1] are webpage networks; Citeseer [55], PubMed [55] and DBLP [5] are paper citation networks; CS [56] is co-authorship network.

For graph classification, we adopt 6 widely adopted datasets (Tab. 8 in Appx. A.2). D&D and PROTEINS [14] are protein networks; IMDB-BINARY and IMDB-MULTI [74] are movie-collaboration datasets; COX2 [60] and NCI109 [64] are molecule datasets.

Baselines. For node classification, we adopt competitive baselines from three categories: 1) 2-layer MLP; 2) Manual MPNNs: GCN [30], GAT [63], and SAGE [24] are classic 1*hop* MPNNs; GBK-GNN [15] develop signed receptive field; SGC [70] directly deploys high-order neighbors as receptive field; ChebNet [11] and H_2GCN [83] jointly utilize different orders of neighbors; WRGAT [59] design graph-rewiring technique that leverages structural properties for multi-channel receptive field; APPNP [31] and GPRGNN [9] are DMPNNS with adaptive receptive fields fusion; 3) AutoMPNNs: AutoGEL [67] explicitly models edge information in graphs.

For the graph classification task, we compare with following baselines: 1) Manually-designed MPNNs: GCN [30], GAT [63], SAGE [24], GIN [72], JKNet [73], and DGCNN [77] with global graph pooling; ASAP [52], SAGPool [35], Graph U-Net [21], and DiffPool [75] with hierarchical graph pooling. 2) AutoMPNNs: GraphNAS [22] is the pioneering work of AutoMPNNs; PAS [69] focuses on pooling operator search; and AutoGEL [67].

Implementation details. MpnnDRF [2] is implemented based on PyTorch [49] and PyTorch Geometric [19]. All experiments are conducted using one single NVIDIA Tesla V100 GPU. Effectiveness results are reported as the average accuracy (with standard derivation) over 30 different runs to eliminate randomness. Baseline results are derived by running their official implementations.

4.2 Main Results

4.2.1 Comparison on the node classification task. The model comparisons on the node classification task are summarized in Tab. 4. 1*hop* MPNNs (GCN, GAT, SAGE) improve simplest MLP on homophilic graphs by leveraging useful information from first-order receptive field, but they cannot handle heterophilic graphs (first 4 graphs). The signed receptive field (GBK-GNN) to jointly model feature similarities and dissimilarities brings gains over class 1*hop* MPNNs (with positive sign for receptive field) due to better flexibility. Directly deploying high-order neighbors as receptive field (SGC) seems not an adequate choice since it always underperforms 1*hop* MPNNs. The combination of high-order and first-order neighbors (ChebNet, H_2GCN) yields consistently better results than their separate usage across all data sets. The improvements are more prominent on heterophilic graphs where useful messages may be non-adjacent, but they still fail to beat the simplest MLP, indicating that the over-squashing issue compromises model performance in this case. Degree centrality based re-wiring that jointly captures proximity and structural information in graphs (WRGAT) demonstrates similar capability with other multi-channel MPNNs or multi-order models. Decoupled and adaptive MPNNs (APPNP, GPRGNN) are strong baselines, which eliminate the potential negative impact of over-sized model parameters and benefit from a larger receptive field without over-smoothing. Meanwhile, they also achieve comparable or better results on heterophilic graphs. AutoMPNN (AutoGEL) that automates the designs of message aggregation and transformation demonstrates significantly superior results over manual MPNNs. Moreover, MpnnDRF not only achieves consistently superior and robust test performances as shown in Tab. 4, but also overall high *MAD* values (0.53-0.68) and high I_{Group} values (0.40-0.79) when dealing with different graph benchmarks and MPNN layer settings (2-10). It indicates the great capacity of MpnnDRF to better capture receptive fields and alleviate well-known issues, including over-smoothing and over-squashing.

Moreover, computational time may be one of the major concerns of AutoMPNNs (including MpnnDRF). To alleviate it, we follow DMPNN to eliminate non-linear transformation matrix for more light-weighted MPNNs; besides, we employ the popular weight-sharing mechanism [51] to avoid training hundreds of candidate MPNNs to convergence, thus further boosting efficiency. We show empirical computational efficiency of MpnnDRF and several baselines in Tab. 6, where search-based MpnnDRF does not significantly increases running time over non-searching baselines. More discussions and results in terms of efficiency and scalability are provided in Appx. A.4.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb
[2] Code is available at https://github.com/zwangeo/MpnnDRF

Table 5: Model performance (accuracy with standard derivation) on the graph classification task.

Type	Model	D&D	PROTEINS	IMDB-B	IMDB-M	COX2	NCI109
Manual MPNNs	GCN	78.12 (4.33)	74.84 (2.82)	72.67 (6.42)	50.40 (3.02)	79.23 (2.19)	73.44 (1.92)
	GTA	75.56 (3.72)	75.30 (3.72)	74.07 (4.53)	49.67 (4.30)	81.56 (4.17)	74.10 (2.45)
	SAGE	77.27 (4.06)	73.75 (2.97)	72.17 (5.29)	48.53 (5.43)	80.31 (5.94)	75.53 (1.64)
	GIN	75.40 (3.68)	74.48 (2.78)	71.67 (2.77)	49.80 (2.50)	83.09 (4.17)	74.56 (2.10)
	JKNet	77.69 (2.35)	75.39 (5.17)	73.47 (4.45)	48.86 (4.96)	79.66 (2.26)	73.83 (1.88)
	DGCNN	76.66 (4.03)	73.57 (4.69)	73.67 (5.70)	49.00 (3.56)	79.85 (2.64)	75.06 (1.65)
	ASAP	77.35 (4.15)	74.93 (3.57)	74.27 (3.97)	50.13 (3.44)	80.95 (3.20)	73.76 (2.24)
	SAGPool	75.06 (5.06)	73.12 (4.47)	74.87 (4.09)	49.33 (4.90)	79.45 (2.98)	64.89 (3.15)
	Graph U-Net	77.10 (5.17)	74.40 (3.49)	73.17 (4.84)	48.80 (3.19)	80.30 (4.21)	72.79 (2.29)
	DiffPool	77.75 (4.00)	73.55 (3.22)	71.86 (5.63)	49.53 (3.98)	79.66 (2.64)	73.15 (2.14)
Auto MPNNs	GraphNAS	71.98 (4.54)	72.51 (3.36)	71.10 (2.30)	46.93 (3.64)	77.73 (1.40)	72.28 (2.28)
	PAS	78.96 (3.68)	76.64 (3.29)	75.10 (5.32)	52.20 (3.73)	83.44 (6.33)	76.84 (2.72)
	AutoGEL	81.31 (2.94)	78.29 (3.11)	74.85 (3.12)	52.41 (2.85)	85.07 (5.32)	**78.68 (1.53)**
	MpnnDRF	**82.74 (2.19)**	**81.64 (2.79)**	**76.56 (1.61)**	**54.07 (2.79)**	**89.04 (2.30)**	77.59 (1.05)

Table 6: Computational time (clock time in seconds, running on one single Tesla V100) comparison with competitive baselines on the node classification task.

Model	Acto.	Corn.	Texa.	Wisc.	Cite.	PubM.	DBLP	CS
SGC	15	11	12	20	204	219	260	407
ChebNet	21	32	30	29	47	48	36	87
GPRGNN	25	23	21	24	36	37	34	65
MpnnDRF	31	24	26	28	49	71	69	91

4.2.2 Comparison on the graph classification task. Furthermore, we demonstrate the performance on the graph classification task in Tab. 5. We observe that on the selected benchmarks, performance gains brought by more sophisticated graph pooling function $read(\cdot)$ are only marginal compared to simple global pooling. Instead, it may produce more promising results by carefully designing more appropriate receptive fields (MpnnDRF) and manipulating messages (MpnnDRF and AutoGEL) during the iterative stage.

4.3 Ablation Studies.

Apart from main results, we further propose variants with degraded space (see Tab. 9) and conduct ablation studies to investigate the influence of MpnnDRF's important design choices regarding receptive field (see Sec. 3.3): 1) the receptive field expansion mechanism, 2) the attentive receptive field, 3) receptive field fusion, 4) decoupled schema to aggregate and transform messages from receptive field, and 5) PE/SE-based methods. MpnnDRF-1*hop* uses the fixed 1*hop* expansion mechanism; MpnnDRF-$\backslash att$ only allows basic degree-based normalization; MpnnDRF-$\backslash inter\_agg$ is restricted to non-adaptive receptive fusion; MpnnDRF-*couple* follows the coupled schema; MpnnDRF-*PE*/*SE_rewire* and MpnnDRF-*PE*/*SE_weight* leverage PE/SE to re-wiring neighbors to induce receptive field and define weighted receptive field respectively. Empirical results of variants on node classification task are provided in Tab. 10 and 11.

We notice that MpnnDRF-1*hop* is a strong baseline on several homophilic data sets but inferior tha MpnnDRF and other variants on heterophilic graphs, indicating more advanced neighbor mechanism is indeed crucial MPNNs, especially for heterophilic graphs. However, even with the fixed 1hop mechanism, MpnnDRF-1*hop* achieves at least comparable results with selected manual MPNNs, implying that MpnnDRF's other design dimensions may work as supplemental when the most suitable expansion mechanism is not available. Similar patterns are exhibited in MpnnDRF-$\backslash att$ and MpnnDRF-$\backslash inter\_agg$, but MpnnDRF-$\backslash att$ suffers from slightly more severe degrades on heterophilic graphs and MpnnDRF-$\backslash inter\_agg$ on homophilic graphs respectively, which validates that different graphs may depend more on different designs. MpnnDRF-*couple* requires $O(Kd^2)$ more model parameters but fail to bring gains over MpnnDRF no matter different graph properties. This also aligns with findings from previous works [31] that the nonlinear transformation may not be the critical and necessary design for MPNNs.

5 CONCLUSION

In this paper, we present a novel MpnnDRF to improve MPNNs' ability by capturing the receptive field in different graphs. We first investigate existing solutions to capture receptive field, then propose improvements in intra-layer inter-layer design dimensions. To handle data-dependent properties, we incorporate existing designs into a search space and formulate a search framework, which can search suitable MPNNs to capture the receptive field for the given graph data. On node classification and graph classification task, MpnnDRF achieves consistently leading results than baselines across popular graph benchmarks. For the future works, it may be worth trying to investigate the discussed problems in other graph tasks, like link prediction on knowledge graphs [12, 13, 57].

ACKNOWLEDGMENTS

Lei Chen's work is partially supported by National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C2004-21GF, AOE Project AoE/E-603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab grant and HKUST Global Strategic Partnership Fund (2021 SJTU-HKUST). Shimin Di's work is supported by the JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In international conference on machine learning. PMLR, 21–29.

[2] Uri Alon and Eran Yahav. 2020. On the bottleneck of graph neural networks and its practical implications. arXiv preprint arXiv:2006.05205 (2020).

[3] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. 2021. Directional graph networks. In International Conference on Machine Learning. PMLR, 748–758.

[4] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 3950–3957.

[5] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017).

[6] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2464–2473.

[7] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 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, Vol. 34. 3438–3445.

[8] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2020. Scalable graph neural networks via bidirectional propagation. Advances in neural information processing systems 33 (2020), 14556–14566.

[9] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive universal generalized pagerank graph neural network. arXiv preprint arXiv:2006.07988 (2020).

[10] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. Noise reduction in speech processing (2009), 1–4.

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems 29 (2016).

[12] Shimin Di and Lei Chen. 2023. Message Function Search for Knowledge Graph Embedding. In Proceedings of the ACM Web Conference 2023. 2633–2644.

[13] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to sparsify tensor decomposition for n-ary relational data. In Proceedings of the Web Conference 2021. 4043–4054.

[14] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. Journal of molecular biology 330, 4 (2003), 771–783.

[15] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. 2022. GBK-GNN: Gated Bi-Kernel Graph Neural Networks for Modeling Both Homophily and Heterophily. In Proceedings of the ACM Web Conference 2022. 1550–1558.

[16] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699 (2020).

[17] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking graph neural networks. arXiv preprint arXiv:2003.00982 (2020).

[18] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2021. Graph neural networks with learnable structural and positional representations. arXiv preprint arXiv:2110.07875 (2021).

[19] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428 (2019).

[20] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. Advances in neural information processing systems 30 (2017).

[21] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In international conference on machine learning. PMLR, 2083–2092.

[22] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. arXiv preprint arXiv:1904.09981 (2019).

[23] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In International conference on machine learning. PMLR, 1263–1272.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. Advances in neural information processing systems 30 (2017).

[25] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 639–648.

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.

[27] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016).

[28] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems 33, 2 (2021), 494–514.

[29] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. 2021. Universal graph convolutional networks. Advances in Neural Information Processing Systems 34 (2021), 10654–10664.

[30] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).

[31] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018).

[32] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. arXiv preprint arXiv:1911.05485 (2019).

[33] Risi Imre Kondor and John Lafferty. 2002. Diffusion kernels on graphs and other discrete structures. In Proceedings of the 19th international conference on machine learning, Vol. 2002. 315–322.

[34] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. Advances in Neural Information Processing Systems 34 (2021), 21618–21629.

[35] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In International conference on machine learning. PMLR, 3734–3743.

[36] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In Proceedings of the IEEE/CVF international conference on computer vision. 9267–9276.

[37] Haoyang LI, Shimin DI, and Lei Chen. 2022. Revisiting Injective Attacks on Recommender Systems. Advances in Neural Information Processing Systems 35 (2022), 29989–30002.

[38] Haoyang Li, Shimin Di, Zijian Li, Lei Chen, and Jiannong Cao. 2022. Black-box Adversarial Attack and Defense on Graph Neural Networks. In 2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, 1017–1030.

[39] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. Advances in Neural Information Processing Systems 33 (2020), 4465–4478.

[40] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In Thirty-Second AAAI conference on artificial intelligence.

[41] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018).

[42] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 338–348.

[43] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is Heterophily A Real Nightmare For Graph Neural Networks To Do Node Classification? arXiv preprint arXiv:2109.05641 (2021).

[44] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016).

[45] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. 2021. Graphit: Encoding graph structure in transformers. arXiv preprint arXiv:2106.05667 (2021).

[46] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. 2022. Transformer for graphs: An overview from architecture perspective. arXiv preprint arXiv:2202.08455 (2022).

[47] Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification. arXiv preprint arXiv:1905.10947 (2019).

[48] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. Technical Report. Stanford InfoLab.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32 (2019).

[50] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020).

[51] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In International conference on machine learning. PMLR, 4095–4104.

[52] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. 2020. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 5470–5477.

[53] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. arXiv preprint arXiv:2004.11198 7 (2020), 15.

[54] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. Learning to simulate complex physics with graph networks. In International Conference on Machine Learning. PMLR, 8459–8468.

[55] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. AI magazine 29, 3 (2008), 93–93.

[56] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018).

[57] DI Shimin, YAO Quanming, Yongqi Zhang, and CHEN Lei. 2021. Efficient relation-aware scoring function search for knowledge graph embedding. In 2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE, 1104–1115.

[58] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Adaptive propagation graph convolutional network. IEEE Transactions on Neural Networks and Learning Systems 32, 10 (2020), 4755–4760.

[59] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. 2021. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. arXiv preprint arXiv:2106.06586 (2021).

[60] Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. Journal of chemical information and computer sciences 43, 6 (2003), 1906–1915.

[61] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 807–816.

[62] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. 2021. Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522 (2021).

[63] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).

[64] Nikil Wale, Ian A Watson, and George Karypis. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems 14, 3 (2008), 347–375.

[65] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. 2022. Equivariant and stable positional encoding for more powerful graph neural networks. arXiv preprint arXiv:2203.00199 (2022).

[66] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining. 1243–1253.

[67] Zhili Wang, Shimin Di, and Lei Chen. 2021. AutoGEL: An Automated Graph Neural Network with Explicit Link Information. Advances in Neural Information Processing Systems 34 (2021), 24509–24522.

[68] Lanning Wei, Huan Zhao, and Zhiqiang He. 2022. Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective. In Proceedings of the ACM Web Conference 2022. 1381–1391.

[69] Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. 2021. Pooling architecture search for graph classification. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2091–2100.

[70] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In International conference on machine learning. PMLR, 6861–6871.

[71] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. SNAS: stochastic neural architecture search. arXiv preprint arXiv:1812.09926 (2018).

[72] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018).

[73] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In International conference on machine learning. PMLR, 5453–5462.

[74] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 1365–1374.

[75] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems 31 (2018).

[76] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. Advances in neural information processing systems 31 (2018).

[77] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In Proceedings of the AAAI conference on artificial intelligence, Vol. 32.

[78] Wentao Zhang, Yu Shen, Zheyu Lin, Yang Li, Xiaosen Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Gmlp: Building scalable and flexible graph neural networks with feature-message passing. arXiv preprint arXiv:2104.09880 (2021).

[79] Wentao Zhang, Yu Shen, Zheyu Lin, Yang Li, Xiaosen Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2022. Pasca: A graph neural architecture search system under the scalable paradigm. In Proceedings of the ACM Web Conference 2022. 1817–1828.

[80] Wentao Zhang, Mingyu Yang, Zeang Sheng, Yang Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Node dependent local smoothing for scalable graph learning. Advances in Neural Information Processing Systems 34 (2021), 20321–20332.

[81] Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2021. Automated machine learning on graphs: A survey. arXiv preprint arXiv:2103.00742 (2021).

[82] Hao Zhu and Piotr Koniusz. 2020. Simple spectral graph convolution. In International Conference on Learning Representations.

[83] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. Advances in Neural Information Processing Systems 33 (2020), 7793–7804.

Table 10: Performance (accuracy with standard derivation) of MpnnDRF variants on node classification for homo. data.

Model	Cite.	PubM.	DBLP	CS
MpnnDRF	**81.08 (0.74)**	**90.47 (0.44)**	**92.69 (1.47)**	**96.06 (0.55)**
MpnnDRF-1*hop*	80.63 (0.84)	90.27 (0.55)	91.91 (1.24)	95.85 (0.33)
MpnnDRF-*att*	80.83 (0.39)	89.76 (0.50)	92.52 (1.37)	95.79 (0.26)
MpnnDRF-*inter_agg*	78.68 (1.77)	90.05 (0.62)	90.59 (1.40)	95.57 (0.35)
MpnnDRF-*couple*	79.72 (0.97)	89.67 (0.64)	92.05 (1.66)	95.79 (0.36)

Table 11: Performance (accuracy with standard derivation) of MpnnDRF variants on node classification for heter. data.

Model	Acto.	Corn.	Texa.	Wisc.
MpnnDRF	42.79 (0.95)	**92.99 (2.62)**	**95.08 (1.20)**	**94.83 (1.89)**
MpnnDRF-1*hop*	40.74 (2.19)	86.38 (5.80)	90.16 (2.32)	86.44 (6.19)
MpnnDRF-*PE/SE_rewire*	41.82 (1.73)	90.53 (4.09)	93.01 (2.55)	90.56 (3.99)
MpnnDRF-*att*	41.86 (1.33)	90.96 (4.61)	91.90 (2.28)	89.12 (9.35)
MpnnDRF-*PE/SE_weight*	42.05 (1.40)	91.23 (3.94)	92.61 (2.31)	90.95 (3.77)
MpnnDRF-*inter_agg*	42.75 (1.07)	90.00 (3.57)	91.39 (3.35)	92.38 (2.85)
MpnnDRF-*couple*	**43.16 (1.24)**	92.66 (3.59)	93.75(2.91)	93.81 (2.78)

Table 7: Data statistics for node classification task.

Property	Acto.	Corn.	Texa.	Wisc.	Cite.	PubM.	DBLP	CS
homo. ratio	0.22	0.13	0.11	0.20	0.74	0.80	0.83	0.81
# Nodes	7600	183	183	251	3327	19717	17716	18333
# Edges	26705	278	287	458	4552	44324	52867	81894
# Features	932	1703	1703	1703	3703	500	1639	6805
# Classes	5	5	5	5	6	3	4	15

Table 8: Data statistics for graph classification task.

Property	D&D	PROTEINS	IMDB-B	IMDB-M	COX2	NCI109
Domain	Bioinfo.	Bioinfo.	Social	Social	Chem.	Chem.
# Graphs	1718	1113	1000	1500	467	4127
# Features	89	3	0	0	3	0
# Classes	2	2	2	3	2	2
Avg. # Nodes	384.3	39.1	19.8	13	41.2	26.69
Avg. # Edges	715.7	72.8	96.5	65.9	43.5	32.13

Table 9: Summary of MpnnDRF variants.

Model	Degraded Space
MpnnDRF	-
MpnnDRF-1*hop*	$\bar{N}(v) = \{1hop\}$
MpnnDRF-*PE/SE_rewire*	$\bar{N}(v) = \{PE\_rewire, SE\_rewire\}$
MpnnDRF-*att*	$\mathbf{e}_{uv} = \{non\_norm, sys\_norm, rw\_norm\}$
MpnnDRF-*PE/SE_weight*	$\mathbf{e}_{uv} = \{rel\_lepe, rel\_rwpe\}$
MpnnDRF-*inter_agg*	$layer\_agg = \{concat, mean\}$
MpnnDRF-*couple*	$upd = \{\sigma\mathbf{W}\}$

A SUPPLEMENT

A.1 Over-smoothing and -squashing: Existing Metrics

A.1.1 MAD for Over-smoothing. Mean Average Distance [7, 42]: $MAD = \frac{1}{|V|^2}\sum_{u,v\in V} dist(\mathbf{H}_u, \mathbf{H}_v)$ (l_p-norm or cosine distance are common choices for $dist(\cdot)$) is one of the popular metrics for measuring the over-smoothing issue. MAD falls within the range of $[0, 1]$, and $MAD \to 0$ may indicate the occurrence of over-smoothing.

A.1.2 Jacobian for Over-squashing. On heterophilic graph data sets (which are prevalent in the real world), over-squashing may become another prominent issue. It refers to MPNNs' inability to propagate distant messages without severe distortion due to the bottleneck originating from the original graph topology [2, 62]. Jacobian [62, 73] is an existing metric to study over-squashing, defined as $I(v,u) = \left|\frac{\partial \mathbf{H}_v}{\partial \mathbf{X}_u}\right|$. It measures the absolute influence (sensitivity) of node u to node v, i.e., on what extent the output representation of v would be influenced by the input attribute of u. When u is important but distant for given v, $I(v,u) \to 0$ may indicate the occurrence of over-squashing for the message from node u.

A.2 Dataset Statistics

We follow common practice to transform directed benchmark graphs Actor, Cornell, Texas, and Wisconsin into undirected graphs and leverage 60%/20%/20% nodes per class as training/validation/testing samples for the node classification task [9]; for graph classification, we employ 10-fold validation [69].

A.3 MpnnDRF Variants: Summary and Results.

See Tab. 9, 10, and 11.

A.4 Computational Efficiency and Scalability

MpnnDRF makes following efforts to guarantee the computational overhead. Firstly, MpnnDRF follows the popular weight-sharing mechanism in Neural Architecture Search (NAS) community to build its super-net, which forces all candidate MPNNs within the super-net to share weights (referred as one-shot NAS because it only needs to train the network weights once). Such way can avoid training hundreds of candidate MPNNs to convergence, thereby saving computational costs. Secondly, MpnnDRF follows advanced DMPNN to design its message passing functions (see Eq. (10), (11), and (12)). It eliminates the non-linear transformation matrix at each layer, thus further reducing the computational cost. Empirically, we further provide concrete computational time of MpnnDRF and several baselines on node classification in Tab. 6, where searching-based MpnnDRF does not require significantly longer running time compared with non-searching MPNN baselines.

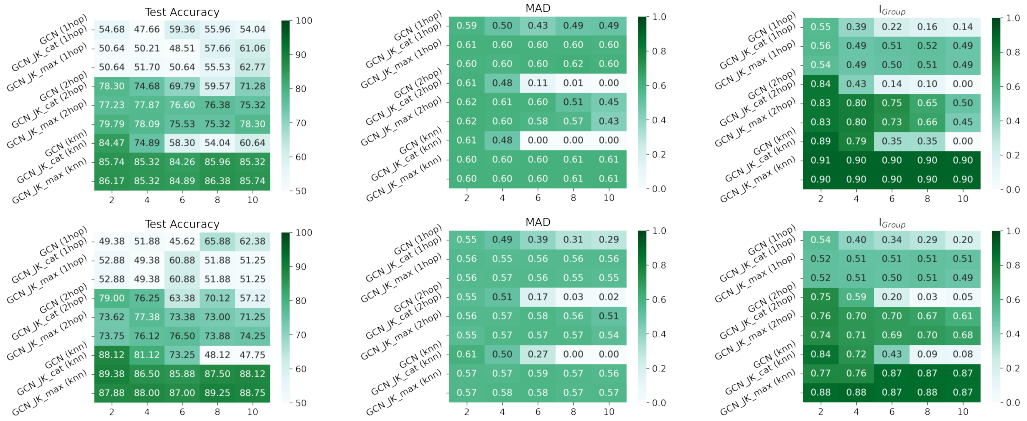
For scalability, we additionally evaluate MpnnDRF on more scale dataset ogbg-molhiv (contains more than 41K graphs) and AUC results are shown in Tab. 12. MpnnDRF achieves better or at least comparable results compared with competitive baselines, indicating the efficacy of MpnnDRF to handle more scale dataset. Moreover, we further present concrete computational time of MpnnDRF on adopted graph classification benchmarks as in Fig. 13. We observe that MpnnDRF exhibits nice scalability with the increase of dataset scale (see Tab. 8 for more details about dataset scale and statistics).

Table 12: Model performance (AUC with standard derivation) on the ogbg-molhiv dataset for graph classification task.

Model	GCN	GIN	PNA	SAN	MpnnDRF
molhiv	76.06 (0.97)	75.58 (1.40)	79.05 (1.32)	77.85 (0.25)	79.31 (1.21)

Table 13: Computational time (clock time in hours, running on one single Tesla V100) of the proposed MpnnDRF on different scale of datasets for the graph classification task.

Model	D&D	PROTEINS	IMDB-B	IMDB-M	COX2	NCI109	molhiv
MpnnDRF	1.93	0.74	0.66	0.55	0.40	0.47	3.91



(a) Test accuracy (b) *MAD* (c) I_{Group}

Figure 4: Results of GCN variants in terms of different neighbor mechanisms and *inter_agg* on Cornell and Wisconsin dataset.