Towards Dynamic Message Passing on Graphs

Junshu Sun[1,2] **Chenxue Yang**[3] **Xiangyang Ji**[4] **Qingming Huang**[1,2,5] **Shuhui Wang**[1,5*]
[1]Institute of Computing Technology, CAS [2]University of Chinese Academy of Sciences
[3]Agriculture Information Institute, CAAS [4]Tsinghua University [5]Peng Cheng Laboratory
{sunjunshu21s,wangshuhui}@ict.ac.cn
yangchenxue@caas.cn xyji@tsinghua.edu.cn qmhuang@ucas.ac.cn

Abstract

Message passing plays a vital role in graph neural networks (GNNs) for effective feature learning. However, the over-reliance on input topology diminishes the efficacy of message passing and restricts the ability of GNNs. Despite efforts to mitigate the reliance, existing study encounters message-passing bottlenecks or high computational expense problems, which invokes the demands for flexible message passing with low complexity. In this paper, we propose a novel dynamic message-passing mechanism for GNNs. It projects graph nodes and learnable pseudo nodes into a common space with measurable spatial relations between them. With nodes moving in the space, their evolving relations facilitate flexible pathway construction for a dynamic message-passing process. Associating pseudo nodes to input graphs with their measured relations, graph nodes can communicate with each other intermediately through pseudo nodes under linear complexity. We further develop a GNN model named $\mathbf{N^2}$ based on our dynamic message-passing mechanism. $\mathbf{N^2}$ employs a single recurrent layer to recursively generate the displacements of nodes and construct optimal dynamic pathways. Evaluation on eighteen benchmarks demonstrates the superior performance of $\mathbf{N^2}$ over popular GNNs. $\mathbf{N^2}$ successfully scales to large-scale benchmarks and requires significantly fewer parameters for graph classification with the shared recurrent layer.

1 Introduction

The inherent irregular structure of graphs poses nontrivial challenges in graph learning [77]. To enable effective learning on graphs, graph neural networks (GNNs) [33, 64] have been specifically designed for graph-structured data. Within GNN models, message passing serves as a crucial function in extracting informative graph features [22, 63]. The vanilla message passing [33, 66] applies node-centric aggregation constrained between the adjacent nodes. During this aggregation process, central nodes access their neighbors in an isotropic manner [29] and aggregate multi-hop features iteratively. In consequence, the vanilla message passing relies heavily on the input graph structure, leading to over-smoothing [46] or over-squashing [2, 14, 15] issues on GNNs.

To further improve the effectiveness of GNNs for graph representation learning, one straightforward solution is to

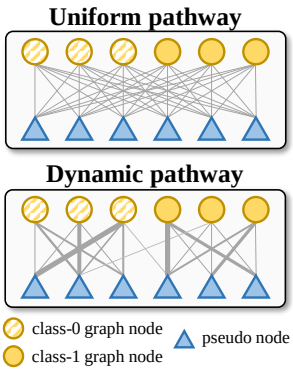


Figure 1: **Comparison between different connection patterns for pseudo nodes and graph nodes.**

*Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-2572

decouple message pathways from the input graph structure. Following this direction, methods [19, 12] have been proposed to perform message passing beyond input structures for global information exchange. Some methods directly model the pairwise relation between nodes [72, 35, 43], but the dense relation causes high computational and space complexity. Other methods incorporate pseudo nodes connected with all the graph nodes to serve as message pathways [22, 30, 42]. However, these models employ uniform pathways, *i.e.*, each pseudo node is connected to all the graph nodes with equal weights, as illustrated at the top of Fig. 1. In consequence, an overwhelming number of node features are squashed equally into pseudo nodes, leading to information bottlenecks on pseudo nodes for message passing and less discriminative representations for downstream tasks [57].

The above limitations call for message passing with flexible pathways and low complexity. In this paper, we propose a novel dynamic message-passing mechanism. To construct flexible pathways, our method measures the specific spatial relations between nodes across time and gives rise to dynamic pathways, as illustrated at the bottom of Fig. 1. To reduce complexity, learnable pseudo nodes are introduced as message-passing proxies between pairs of graph nodes.

Specifically, both graph nodes and pseudo nodes are embedded in a common space with measurable spatial relations between them. By moving nodes in the space, their measured relations evolve accordingly, facilitating a dynamic message-passing process with flexibility. Regarding the measured relation as pseudo edges, the message passing on the input graphs can be extended to pseudo nodes. As a result, graph nodes can communicate with each other intermediately through pseudo nodes, free from dense relation modeling.

To achieve this dynamic process, we further develop a GNN model named $\mathbf{N^2}$, based on our graph **N**odes and pseudo **N**odes mechanism for message passing. $\mathbf{N^2}$ incorporates a recurrent layer to parameterize the displacements of graph nodes and pseudo nodes in the common space. With both types of nodes moving in the common space, $\mathbf{N^2}$ measures the actively changing spatial relations and constructs evolving pathways for dynamic message passing.

Our contributions are summarized as follows. First, we design a flexible and low-complexity message-passing mechanism from a new perspective, where dynamic message pathways are built upon evolving spatial relations between nodes. Second, we develop a novel GNN model named $\mathbf{N^2}$ to achieve dynamic message passing, which employs a recurrent layer to parameterize the evolutionary displacements of nodes. Third, we demonstrate the advantages of $\mathbf{N^2}$ on eighteen real-world benchmarks, where $\mathbf{N^2}$ achieves superior performance. Codes are available at `https://github.com/sunjss/N2`.

2 Related Work

Flexible message passing on graphs. In pursuit of expressive operators for graph learning, methods propose to approximate diverse filters with parameterized polynomials [11, 13, 34]. However, due to the prohibitive computational complexity of higher-order polynomials, these methods [33, 71] are constrained with lower orders and only perform local aggregation on graphs. This constrained process couples message passing with input topology and contains inherent limitations, including over-smoothing [39, 46] and over-squashing [2, 60]. To overcome these limitations, some works try to decouple message passing from input topology and introduce alternate pathways, including edge shortcuts [1, 60, 12, 24], pseudo nodes [42, 57], and graph pooling operations [20, 53, 74].

By adding edge shortcuts, methods aim to improve the message-passing efficiency on graphs. These methods can relieve certain bottlenecks on the input graphs [12, 50, 60] and aggregate multi-hop information during message passing [1, 24]. Notably, we categorize graph structure learning methods [79, 31, 65, 76] into the edge shortcut paradigm, which constructs message pathways by modeling edge connectivity between nodes. While edge shortcuts refine local connections, pseudo nodes directly enable global message passing. However, the pseudo nodes in prior works employ uniform pathways to connect with graph nodes, which become bottlenecks in message passing [57] and limit efficient global communication. Unlike these works, $\mathbf{N^2}$ models dynamic interactions between graph nodes and pseudo nodes, with edge weights varying flexibly across them.

Another line of effort in decoupling message passing from input topology is hierarchical GNNs. These methods [53, 74] learn multi-scale graph features through iterative graph pooling, *i.e.*, node clustering or node drop. Node clustering [73, 3] learns soft assignment matrices to aggregate nodes into coarser

levels. On the other hand, node drop [20, 37] ranks and selects salient subsets to prune less critical nodes. Different from graph pooling methods, $\mathbf{N^2}$ performs both local and global message passing in each recursive step, avoiding information loss in coarser graphs [67].

Reducing complexity for global message passing. Self-attention [62] that models pairwise relations between nodes can be seen as message passing on fully connected graphs. However, dense attention requires quadratic space and computational complexity, which is intractable for large-scale graphs. In order to scale attention-based global message passing to larger graphs, recent methods propose to approximate dense attention through expander graphs [57], kernel functions [69], and diffusion [68]. One similar work [7] to ours proves that message-passing layers with a single pseudo node can approximate dense attention. In this paper, we follow a contrary thread and develop $\mathbf{N^2}$ with a single shared message passing layer and multiple pseudo nodes. Each pseudo node interacts dynamically with graph nodes, avoiding becoming message-passing bottlenecks as the uniform connected pattern [22, 42].

Recurrent layer for graph learning. Scarselli et al. [54] first employ a recurrent layer to update node features recursively. According to Banach's fixed point theorem [4], implementing the recurrent layer as a contraction mapping guarantees the existence of a unique fixed point representation for any input graph, towards which the recursive updates converge. However, their contraction mapping formulation is topology-dependent, incurring over-smoothing as the number of recursive steps increases. In contrast, the recurrent layer in $\mathbf{N^2}$ decouples message passing from input topology, empowering flexible communication between graph nodes.

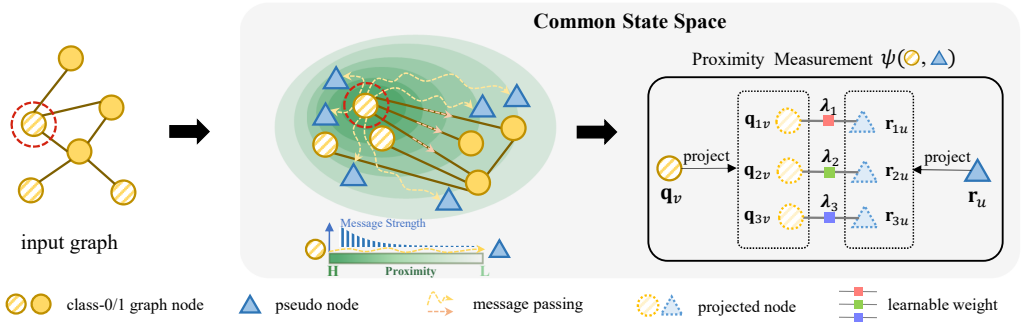


Figure 2: **Dynamic message-passing pathway construction in common state space.** Graph nodes and pseudo nodes interact actively in the common state space, constructing dynamic message pathways through proximity measurement. In empirical model analysis, pseudo nodes tend to be attracted toward a distinct graph node cluster.

3 Towards Dynamic Message Passing

Our dynamic message-passing mechanism parameterizes message pathways with measurable relations between nodes in a common space. The displacements of nodes in the space give rise to evolving relations. As a result, the message-passing process also changes dynamically. This section defines the common space tailored for the dynamic message-passing process, where pseudo nodes are employed to reduce the computational complexity. The next section demonstrates how $\mathbf{N^2}$ learns the displacements of nodes towards the dynamic process.

3.1 Preliminaries

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ denotes the node set of size n and $\mathcal{E} = \{e_{v_i,v_j} | v_j \in \mathcal{N}(v_i)\}$ denotes the edge set of size m. $\mathcal{N}(\cdot)$ denotes the one-hop neighbor set of a given node. Each node $v \in \mathcal{V}$ corresponds to a feature vector $\mathbf{x}_v \in \mathbb{R}^d$ where d is the number of features. Let $\mathbf{X} = (\mathbf{x}_{v_1}, \cdots, \mathbf{x}_{v_n})^\top \in \mathbb{R}^{n \times d}$ be the node feature matrix composed of feature vectors. Let $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ be the edge feature matrix with d_e features. $\mathbf{E}_{i,j,\cdot} = \mathbf{e}_{v_i,v_j} \in \mathbb{R}^{d_e}$.

Parameterize pathways as shared functions. When introducing pseudo nodes for message passing, uniform pathways [22, 42] directly parameterize the edge weights between pairs of graph nodes and

https://doi.org/10.52202/079017-2572

pseudo nodes. Specifying different weights for each pair entails a parameter count scaling to the number of graph nodes. To ensure a tractable number of parameters for various scales of input graphs, we gain inspiration from DyN [48], a distinctive neural network model that directly models neurons, instead of learning connection weights as conventional approaches [36, 62]. The connection weights in DyN are parameterized with a path integral function given the spatial coordinates of neurons. By sharing the integral function across neurons, DyN circumvents the need for heavy parameters. Drawing an analogy between neurons and nodes, we can also parameterize relations between nodes as a shared function.

3.2 Common State Space

In light of DyN, we propose to unify graph nodes and pseudo nodes in a common state space $\mathcal{S} \subseteq \mathbb{R}^q$ and define their spatial proximity with a shared measurement function. This function measures the relations between pairs of nodes, enabling the construction of dynamic message pathways. Here, we borrow the concept of *state* from prior works, such as DyN and LSTM [26], to denote the learned descriptive embedding of nodes. Examples of the information encoded in the states include node features and local topology. The states of a node identify its spatial coordinate in the state space. For more discussion on "state space", please refer to Appendix A.

Embedded node states. Given a pseudo node set $\mathcal{U} = \{u_1, \cdots, u_{n_p}\}$ of size $|\mathcal{U}| = n_p$, we embed pseudo nodes in the state space $\mathcal{S}$ as learnable parameters $\mathbf{R} = (\mathbf{r}_{u_1}, \cdots, \mathbf{r}_{u_{n_p}})^\top \in \mathbb{R}^{n_p \times q}$. These learnable elements are named "pseudo nodes" to align with "graph nodes" from inputs. Pseudo nodes can be associated with input graphs through pseudo edges and participate in the message passing between graph nodes. We will elaborate on this association after introducing the spatial proximity measurement in the state space. Given the graph node set $\mathcal{V}$ with feature matrix $\mathbf{X}$, we have graph node states $\mathbf{Q} = (\mathbf{q}_1, \cdots, \mathbf{q}_n)^\top = f(\mathbf{X}) \in \mathbb{R}^{n \times q}$, where $f : \mathbb{R}^d \mapsto \mathcal{S}$ is a permutation equivariant function. Displacements of a node in the state space signify shifts in its states, *i.e.,* the learned descriptive embeddings. The objective of a GNN is to model the true distribution of graph nodes in the state space, wherein nodes with similar features obtain proximal embeddings. In the following, we refer to both graph nodes and pseudo nodes collectively as embedded nodes.

Proximity measurement. To model complex relations, we assume that the embedded nodes have a non-linear relationship in the state space $\mathcal{S}$. However, modeling non-linearity via operations such as path integral can be computationally complex. To address this problem, we approximate non-linear relations with piece-wise weighted inner products. Taking proximity measurement between graph nodes as an example, each node is divided into k pieces with their proximity formulated as

$$\psi\left(\mathbf{q}_v, \mathbf{q}_{v'}\right) = \sum_{i=1}^{k} \lambda_i \mathbf{q}_{iv}^\top \mathbf{q}_{iv'}, \quad \mathbf{q}_{i\cdot} = \mathtt{NL}_i(\mathbf{q}_\cdot), \tag{1}$$

where λ_i is a learnable parameter, $\mathtt{NL}(\cdot)$ denotes a non-linear function with linear mapping followed by $\mathtt{LeakyReLU}(\cdot)$. $\psi(\cdot, \cdot)$ is termed as proximity instead of distance because it ranges in $\mathbb{R}$. The spatial proximity is weighted with different inner product similarity between k pairs of pieces to approximate complex relations. We conduct ablation studies on the number of pieces k in Appendix D.5.

3.3 Associating Pseudo Nodes to Input Graphs

Pseudo edges. Based on the measurable proximity between the embedded nodes, we now introduce how to obtain pseudo edges and thus associate pseudo nodes to input graphs. For each pseudo node $u \in \mathcal{U}$ embedded at $\mathbf{r}_u$ in the state space, it has pseudo edges with any pseudo node $u' \in \mathcal{U}$ and graph node $v \in \mathcal{V}$, where pseudo edge weights $e_{u,u'}$ and $e_{u,v}$ can be formulated as the proximity

$$e_{u,u'} = \psi(\mathbf{r}_u, \mathbf{r}_{u'}), \quad e_{u,v} = \psi(\mathbf{r}_u, \mathbf{q}_v). \tag{2}$$

Messages for embedded nodes. Following the common practice in GNNs [63], we interpret the interaction between the embedded nodes as message passing. To achieve this, both graph node v and pseudo node u learn their messages $\mathbf{m}_v, \mathbf{m}_u \in \mathbb{R}^d$ to be passed in the state space. As a result, graph node v and pseudo node u in the state space can be further described as

$$\begin{aligned} u &\triangleq \left(\mathbf{r}_u, \mathbf{m}_u, \{e_{u,w}, e_{w,u} | w \in (\mathcal{U} \cup \mathcal{V})\}\right), \\ v &\triangleq \left(\mathbf{q}_v, \mathbf{m}_v, \{e_{v,w}, e_{w,v} | w \in (\mathcal{N}(v) \cup \mathcal{U})\}\right). \end{aligned} \tag{3}$$

The messages of graph nodes are initialized with node features. Each graph edge $e_{v,v'}$ is characterized by its edge feature $\mathbf{e}_{v,v'}$.

3.4 Dynamic Message Passing in the State Space

Given the measured proximity as message pathways and the messages to be passed, the embedded nodes can interact with each other through message passing. Specifically, the interactions between graph nodes are performed in both local and global scope.

Global message passing. In the state space, graph nodes perform global message passing based on their states, where pseudo nodes serve as proxies. Given graph node states $\mathbf{Q} \in \mathbb{R}^{n\times q}$, pseudo-node states $\mathbf{R} \in \mathbb{R}^{n_p\times q}$ and graph node messages $\mathbf{M}^{\mathtt{n}} = (\mathbf{m}_{v_1}, \cdots, \mathbf{m}_{v_n})^\top \in \mathbb{R}^{n\times d}$, the global message-passing process can be formulated as

$$\texttt{(Diffuse)} \qquad \mathbf{G} = \mathbf{E}^{\mathtt{np}}\mathbf{M}^{\mathtt{n}}, \quad \mathbf{E}^{\mathtt{np}}_{ij} = e_{u_i,v_j} = \psi(\mathbf{R}_{i,\cdot}, \mathbf{Q}_{j,\cdot}), \tag{4}$$

$$\texttt{(Refine)} \qquad \hat{\mathbf{G}} = \mathbf{E}^{\mathtt{pp}}\mathbf{G}, \quad \mathbf{E}^{\mathtt{pp}}_{ij} = e_{u_i,u_j} = \psi(\mathbf{R}_{i,\cdot}, \mathbf{R}_{j,\cdot}), \tag{5}$$

$$\Delta\mathbf{R} = \mathtt{NL}(\hat{\mathbf{G}}), \quad \mathbf{M}^{\mathtt{p}} = \mathtt{NL}(\hat{\mathbf{G}}),$$

$$\texttt{(Collect)} \qquad \mathbf{M}^{\mathtt{glob}} = \mathbf{E}^{\mathtt{pn}}\mathbf{M}^{\mathtt{p}}, \ \mathbf{E}^{\mathtt{pn}}_{ij} = e_{v_i,u_j} = \psi(\mathbf{Q}_{i,\cdot}, [\mathbf{R}+\Delta\mathbf{R}]_{j,\cdot}), \tag{6}$$

where $\mathbf{E}^{\mathtt{np}} \in \mathbb{R}^{n_p\times n}$ denotes the edge weight matrix from graph nodes to pseudo nodes, $\mathbf{E}^{\mathtt{pp}}$ and $\mathbf{E}^{\mathtt{pn}}$ follow the similar name rule. An example of $\mathbf{E}^{\mathtt{np}}_{ij}$ computation is illustrated in Fig. 2. Eq. 4 formulates the process that graph nodes diffuse messages to pseudo nodes. Eq. 5 formulates the global feature refinement at the pseudo-node level, where $\Delta\mathbf{R} \in \mathbb{R}^{n_p\times q}$ denotes the learned displacements for pseudo nodes, $\mathbf{M}^{\mathtt{p}} \in \mathbb{R}^{n_p\times d}$ encodes the refined pseudo-node messages with global information. Eq. 6 formulates the message collection process from pseudo nodes to graph nodes.

For simplicity, we compile Eq. 4-6 as $\mathbf{M}^{\mathtt{glob}}, \Delta\mathbf{R} = \mathtt{GlobMP}(\mathbf{Q}, \mathbf{M}^{\mathtt{n}}, \mathbf{R})$. Note that the space complexity of our global message passing is $O(knn_p)$ with $k, n_p \ll n$, significantly lower than $O(n^2)$ in dense global message passing.

Local message passing. Topology-coupled message passing is employed to encode the local structure. The resulted local message-passing process for graph node v can be formulated as

$$\mathbf{m}_v^{\mathtt{local}} = \frac{1}{|\mathcal{N}(v)|+1}\left[\mathbf{m}_v + \sum_{v'\in\mathcal{N}(v)} \mathtt{NL}(\mathbf{m}_{v'}||\mathbf{e}_{v,v'})\right], \tag{7}$$

where $||$ denotes the concatenate operation. Through local message passing, graph nodes aggregate messages from their adjacent nodes. Eq. 7 can be compiled as $\mathbf{M}^{\mathtt{local}} = \mathtt{LocalMP}(\mathbf{M}, \mathbf{E})$ for all the graph nodes.

4 Implementing Dynamic Message Passing with $\mathbf{N^2}$

We further develop a GNN model named $\mathbf{N^2}$ to move the embedded nodes to their optimal positions. By feeding the states recursively into a single recurrent layer, $\mathbf{N^2}$ learns the displacements of all the embedded nodes in the state space and updates their positions. The changes in position reshape the spatial relations between the embedded nodes and thus reshape the dynamic message pathways. These dynamically evolving pathways empower $\mathbf{N^2}$ to adapt to the specific positions of the embedded nodes at each recursive step. In this section, we first outline the key process in the l-th recursive step, *i.e.*, pseudo-node adaptation and dynamic message passing, then describe different output fashion for downstream tasks.

4.1 Pseudo-node Adaptation

Pseudo nodes are initialized randomly in the state space. To adapt to specific input graphs, $\mathbf{N^2}$ first diffuses graph node messages to pseudo nodes, adjusting pseudo-node states and corresponding messages accordingly. Given graph node messages $\mathbf{M}^{\mathtt{n}(l-1)}$ ($\mathbf{M}^{\mathtt{n}(0)} = \mathbf{X}$), graph node states $\mathbf{Q}^{(l-1)}$ and pseudo-node states $\mathbf{R}^{(l-1)}$ at the l-th recursive step, the adaptation process can be formulated as

$$\begin{aligned} &\hat{\mathbf{M}}^{\mathtt{glob}(l)}, \Delta\hat{\mathbf{R}}^{(l)} = \mathtt{GlobMP}\left(\mathbf{Q}^{(l-1)}, \mathbf{M}^{\mathtt{n}(l-1)}, \mathbf{R}^{(l-1)}\right), \\ &\hat{\mathbf{R}}^{(l)} = \mathbf{R}^{(l-1)} + \Delta\hat{\mathbf{R}}^{(l)}, \end{aligned} \tag{8}$$

https://doi.org/10.52202/079017-2572

Table 1: **Graph classification results on small-scale benchmarks (measured by accuracy: %).**

	PROTEIN	NCI1	IMDB-B	IMDB-M	COLLAB
#GRAPHS	1,113	4,110	1,000	1,500	5,000
#NODES	39.06	29.87	19.77	13.00	74.49
#EDGES	145.60	64.60	193.10	131.87	4,914.4
#NODE FEATURES	3	37	0	0	0
PATCHY-SAN [45]	$75.00_{\pm 2.51}$	$78.60_{\pm 1.90}$	$71.00_{\pm 2.29}$	$45.23_{\pm 2.84}$	$72.60_{\pm 2.15}$
GCN [33]	$73.24_{\pm 0.73}$	$76.29_{\pm 1.79}$	$73.26_{\pm 0.46}$	$50.39_{\pm 0.41}$	$80.59_{\pm 0.27}$
PG [29]	$76.80_{\pm 3.80}$	$82.80_{\pm 1.30}$	$76.80_{\pm 2.60}$	$53.20_{\pm 3.60}$	$80.90_{\pm 0.80}$
COCN [58]	$76.86_{\pm 0.13}$	$82.89_{\pm 0.19}$	$77.26_{\pm 0.27}$	$56.32_{\pm 0.18}$	$86.15_{\pm 0.10}$
GIN [71]	$73.84_{\pm 4.46}$	$76.62_{\pm 1.80}$	$72.78_{\pm 0.86}$	$48.13_{\pm 1.36}$	$78.19_{\pm 0.63}$
+PSEUDO NODE [42]	$74.11_{\pm 4.12}$	$77.08_{\pm 1.49}$	-	-	-
GRAPHSAGE [25]	$73.48_{\pm 5.66}$	$73.82_{\pm 2.17}$	$68.80_{\pm 4.50}$	$47.60_{\pm 3.50}$	$73.90_{\pm 1.70}$
+PSEUDO NODE [42]	$73.93_{\pm 5.68}$	$74.31_{\pm 2.27}$	-	-	-
DIFFPOOL [73]	$75.62_{\pm 5.17}$	$76.62_{\pm 1.93}$	$73.14_{\pm 0.70}$	$51.31_{\pm 0.72}$	$82.13_{\pm 0.43}$
+PSEUDO NODE [42]	$75.98_{\pm 3.89}$	$77.08_{\pm 1.33}$	-	-	-
TOPKPOOL [20]	$70.48_{\pm 1.01}$	$67.02_{\pm 2.25}$	$71.58_{\pm 0.95}$	$48.59_{\pm 0.72}$	$77.58_{\pm 0.85}$
SAGPOOL [37]	$71.56_{\pm 1.49}$	$67.45_{\pm 1.11}$	$72.55_{\pm 1.28}$	$50.23_{\pm 0.44}$	$78.03_{\pm 0.31}$
STRUCTPOOL [74]	$75.16_{\pm 0.86}$	$78.64_{\pm 1.53}$	$72.06_{\pm 0.64}$	$50.23_{\pm 0.53}$	$77.27_{\pm 0.51}$
SEP [67]	$76.42_{\pm 0.39}$	$79.35_{\pm 0.33}$	$74.12_{\pm 0.56}$	$51.53_{\pm 0.65}$	$81.28_{\pm 0.15}$
GMT [3]	$75.09_{\pm 0.59}$	$76.35_{\pm 2.62}$	$73.48_{\pm 0.76}$	$50.66_{\pm 0.82}$	$80.74_{\pm 0.54}$
$\mathbf{N^2}$ (OURS)	$\mathbf{77.53}_{\pm 1.78}$	$\mathbf{83.52}_{\pm 3.75}$	$\mathbf{79.95}_{\pm 2.46}$	$\mathbf{57.31}_{\pm 2.19}$	$\mathbf{86.72}_{\pm 1.62}$

where $\hat{\mathbf{M}}^{\mathtt{glob}(l)}$ denotes pseudo-node messages that are collected by graph nodes, serving as query signals for different patterns on graphs. $\hat{\mathbf{R}}^{(l)}$ denotes the adjusted pseudo-node states.

4.2 Dynamic Message Passing

$\mathbf{N^2}$ performs both local and global message passing. At the local level, graph nodes exchange their own messages $\mathbf{M}^{\mathtt{n}(l-1)}$, collected messages $\hat{\mathbf{M}}^{\mathtt{glob}(l)}$ and graph node states $\mathbf{Q}^{(l-1)}$:

$$\begin{aligned} &\mathbf{M}^{\mathtt{local}(l)} = \mathtt{LocalMP}\left[\left(\mathbf{M}^{\mathtt{n}(l-1)} \| \hat{\mathbf{M}}^{\mathtt{glob}(l)} \| \mathbf{Q}^{(l-1)}\right), \mathbf{E}\right], \\ &\hat{\mathbf{Q}}^{(l)} = \mathbf{Q}^{(l-1)} + \mathtt{NL}(\mathbf{M}^{\mathtt{local}(l)}). \end{aligned} \tag{9}$$

Through local message passing, graph node messages $\mathbf{M}^{\mathtt{local}(l)}$ are generated in response to the query messages $\hat{\mathbf{M}}^{\mathtt{glob}(l)}$. $\mathbf{N^2}$ then sends the updated messages to global message passing:

$$\begin{aligned} &\mathbf{M}^{\mathtt{glob}(l)}, \Delta\mathbf{R}^{(l)} = \mathtt{GlobMP}\left(\hat{\mathbf{Q}}^{(l)}, \mathbf{M}^{\mathtt{local}(l)}, \hat{\mathbf{R}}^{(l)}\right), \\ &\mathbf{Q}^{(l)} = \hat{\mathbf{Q}}^{(l)} + \mathtt{NL}(\mathbf{M}^{\mathtt{glob}(l)}), \quad \mathbf{M}^{\mathtt{n}(l)} = \mathbf{M}^{\mathtt{n}(l-1)} + \mathbf{M}^{\mathtt{glob}(l)}, \\ &\mathbf{R}^{(l)} = \hat{\mathbf{R}}^{(l)} + \Delta\mathbf{R}^{(l)}. \end{aligned} \tag{10}$$

4.3 Output Module

$\mathbf{N^2}$ updates the states of the embedded nodes recursively with a single recurrent layer. Instead of employing different layers for each recursive step, the associated parameters are shared across steps. After L recursive steps, the embedded nodes now reach their final states $\mathbf{Q}^{(L)}$ and $\mathbf{R}^{(L)}$. $\mathbf{N^2}$ then takes graph node states and pseudo-node states as the learned representation for node-level and graph-level tasks, respectively. For graph classification, $\mathbf{N^2}$ further applies $\mathtt{NL}(\cdot)$ to aggregate the pseudo-node states. To make class predictions, $\mathbf{N^2}$ employs learnable parameter $\mathbf{C} \in \mathbb{R}^{n_c \times q}$ as the states of n_c class nodes and outputs the proximity between the recursive output and the class nodes.

5 Experiment

In this section, we provide empirical evaluation results of $\mathbf{N^2}$ on real-world benchmarks. $\mathbf{N^2}$ is implemented with PyTorch [47] and PyTorch Geometric [18], and trained on a single Nvidia Geforce RTX 4090. The detailed experimental settings are presented in Appendix C.

5.1 Graph Classification

Experimental setups. We adopt six benchmarks including three biochemical datasets (OGB-molpcba [27], PROTEINS [44], NCI1 [44]) and three social network datasets [44] (COLLAB,

Table 2: **Graph classification results on large-scale benchmark OGB-molpcba.**

#Graphs 437,929	#Nodes 26	#Edges 28.1
Methods	Average Precision (%)	#Params (K)
GCN [33]	20.20$_{\pm 0.24}$	565
+pseudo node [27]	24.24$_{\pm 0.34}$	2,017
GIN [71]	22.66$_{\pm 0.28}$	1,923
+pseudo node [27]	27.03$_{\pm 0.23}$	3,374
MPNN	-	-
+pseudo node [7]	28.48$_{\pm 0.26}$	-
GraphTrans [7]	27.61$_{\pm 0.29}$	-
SAN [35]	27.65$_{\pm 0.42}$	5,885
GraphGPS [52]	29.07$_{\pm 0.28}$	9,745
Graphormer [72]	31.39$_{\pm 0.32}$	119,529
$\mathbf{N^2}$ (Ours)	**33.90**$_{\pm 0.73}$	516

Table 3: **Node classification results on small-scale homophilic graphs (measured by accuracy: %).**

	CoauthorCS	CoauthorPhy	AmzPhoto	AmzComputers
#Nodes	18,333	34,493	7,487	13,381
#Edges	81,894	247,962	119,043	245,778
GCN [33]	92.92$_{\pm 0.12}$	96.18$_{\pm 0.07}$	92.70$_{\pm 0.20}$	89.65$_{\pm 0.52}$
GAT [64]	93.61$_{\pm 0.14}$	96.17$_{\pm 0.08}$	93.87$_{\pm 0.11}$	90.78$_{\pm 0.17}$
GPRGNN [11]	95.13$_{\pm 0.09}$	96.85$_{\pm 0.08}$	94.49$_{\pm 0.14}$	89.32$_{\pm 0.29}$
APPNP [21]	94.49$_{\pm 0.07}$	96.54$_{\pm 0.07}$	94.32$_{\pm 0.14}$	90.18$_{\pm 0.17}$
GT [56]	94.64$_{\pm 0.13}$	97.05$_{\pm 0.05}$	94.74$_{\pm 0.13}$	91.18$_{\pm 0.17}$
Graphormer [72]	OOM	OOM	92.74$_{\pm 0.14}$	OOM
SAN [35]	94.51$_{\pm 0.15}$	OOM	94.86$_{\pm 0.10}$	89.83$_{\pm 0.16}$
GraphGPS [52]	93.93$_{\pm 0.12}$	OOM	95.06$_{\pm 0.13}$	OOM
NAGphormer [9]	95.75$_{\pm 0.09}$	97.34$_{\pm 0.03}$	95.49$_{\pm 0.11}$	91.22$_{\pm 0.14}$
Exphormer [57]	**95.77**$_{\pm 0.15}$	97.16$_{\pm 0.13}$	95.27$_{\pm 0.42}$	91.59$_{\pm 0.31}$
$\mathbf{N^2}$ (Ours)	94.44$_{\pm 0.45}$	**97.56**$_{\pm 0.28}$	**95.75**$_{\pm 0.34}$	**92.51**$_{\pm 0.13}$

IMDB-BINARY and IMDB-MULTI). The benchmark statistics are summarized in Tab. 1 and Tab. 2. We choose convolutional GNNs, GNNs with a single pseudo node, hierarchical GNNs, and graph transformers as the baselines of graph classification. For more details, please refer to Appendix. C.1.2.

Performance. $\mathbf{N^2}$ and baselines are evaluated on both small-scale and large-scale benchmarks. The evaluation results in Tab. 1 and Tab. 2 showcase the ability of $\mathbf{N^2}$ to outperform various GNNs and graph transformers. Especially on the large-scale benchmark OGB-molpcba, $\mathbf{N^2}$ surpasses baseline models with only 500K parameters, while Graphormer reaches 31.39% average precision with 119.5M parameters. Compared to GNNs with a single pseudo node, $\mathbf{N^2}$ gains significant improvements. This demonstrates the effectiveness of our common state space where pseudo nodes and graph nodes can interact dynamically with each other.

5.2 Node Classification

Experimental setups. For node classification, we conduct experiments on (**1**) six small-scale benchmarks: homophilic graphs (AmazonPhoto, AmazonComputers, CoauthorCS, and CoauthorPhysics) [55], heterophilic graphs (questions, amazon-ratings, tolokers, and minesweeper) [49]; and (**2**) four large-scale benchmarks: homophilic graphs (OGB-arXiv, OGB-proteins) [27], heterophilic graphs (arXiv-year, genius) [41]. The statistics of node classification benchmarks are summarized in Tab. 3-5. We choose convolutional GNNs and graph transformers as the baselines. For detailed baseline setups, please refer to Appendix. C.2.2.

Table 4: **Node classification results on small-scale heterophilic graphs (measured by ROC-AUC except accuracy for amazon-ratings: %).** † denotes our reproduced results.

	Questions	Amazon-ratings	Tolokers	Minesweeper
#Nodes	48,921	24,492	11,758	10,000
#Edges	153,540	93,050	519,000	39,402
SGC [66]	75.91$_{\pm 0.96}$	50.66$_{\pm 0.48}$	80.70$_{\pm 0.97}$	70.88$_{\pm 0.90}$
GCN [33]	76.09$_{\pm 1.27}$	48.70$_{\pm 0.63}$	83.64$_{\pm 0.67}$	89.75$_{\pm 0.52}$
GAT [64]	77.43$_{\pm 1.20}$	49.09$_{\pm 0.63}$	83.70$_{\pm 0.47}$	92.01$_{\pm 0.68}$
GPRGNN [11]	55.48$_{\pm 0.91}$	44.88$_{\pm 0.34}$	72.94$_{\pm 0.97}$	86.24$_{\pm 0.61}$
H_2GCN [78]	63.59$_{\pm 1.46}$	36.47$_{\pm 0.23}$	73.35$_{\pm 1.01}$	89.71$_{\pm 0.31}$
FAGCN [5]	77.24$_{\pm 1.26}$	44.12$_{\pm 0.30}$	77.75$_{\pm 1.05}$	88.17$_{\pm 0.73}$
GloGNN [40]	65.74$_{\pm 1.19}$	36.89$_{\pm 0.14}$	73.39$_{\pm 1.17}$	51.08$_{\pm 1.23}$
GT [56]	77.95$_{\pm 0.68}$	**51.17**$_{\pm 0.66}$	83.23$_{\pm 0.64}$	91.85$_{\pm 0.76}$
Graphormer [72]	OOM	OOM	OOM	OOM
GraphGPS [52]	OOM	OOM	84.70$_{\pm 0.56}$	92.29$_{\pm 0.61}$
Exphormer [57] †	73.86$_{\pm 0.58}$	49.36$_{\pm 0.36}$	84.20$_{\pm 0.22}$	90.42$_{\pm 0.10}$
$\mathbf{N^2}$ (Ours)	**78.07**$_{\pm 0.63}$	50.25$_{\pm 0.53}$	**86.25**$_{\pm 0.41}$	**93.97**$_{\pm 0.27}$

Performance. The evaluation results are presented in Tab. 3-5. $\mathbf{N^2}$ reaches superior or comparable performance against strong GNN baselines on both small-scale and large-scale benchmarks. Compared to graph transformers based on dense attention, including Graphormer, SAN, and GraphGPS, $\mathbf{N^2}$ surpasses the baselines on small-scale benchmarks and successfully scales to large-scale benchmarks. Note that Exphormer based on sparse attention also suffers from the out-of-memory problem in our experimental environment. For other sparse graph transformers, $\mathbf{N^2}$ can achieve comparable results and perform consistently between heterophilic and homophilic benchmarks.

https://doi.org/10.52202/079017-2572

Table 5: **Node classification results on large-scale benchmarks (measured by ROC-AUC/accuracy: %).** † denotes our reproduced results.

METRICS	GENIUS ROC-AUC	ARXIV-YEAR ACCURACY	OGB-ARXIV ACCURACY	OGB-PROTEINS ROC-AUC
#NODES	421,961	169,343	169,343	132,534
#EDGES	984,979	1,166,243	1,166,243	39,561,252
SGC [66]	82.36±0.37	32.83±0.13	67.79±0.27	70.31±0.23
GCN [33]	87.42±0.37	46.02±0.26	71.74±0.29	72.51±0.35
GAT [64]	55.80±0.87	46.05±0.51	67.63±0.23	74.63±1.24
APPNP [21]	85.36±0.62	38.15±0.26	-	-
H_2GCN [78]	OOM	49.09±0.10	OOM	OOM
LINKX [41]	**90.77**±0.27	56.00±1.34	-	-
GRAPHORMER [72]				
SAN [35]		OOM		
GRAPHGPS [52]				
EXPHORMER [57]	OOM	OOM	72.44±0.28	OOM
NODEFORMER [69]	88.62±0.27 (†)	37.68±0.30 (†)	59.90±0.42	77.45±1.15
SGFORMER [70]	83.91±2.60 (†)	44.34±0.07 (†)	**72.63**±0.13	79.53±0.38
$\mathbf{N^2}$ (OURS)	89.32±0.26	**58.69**±0.42	70.01±0.65	**79.55**±0.79

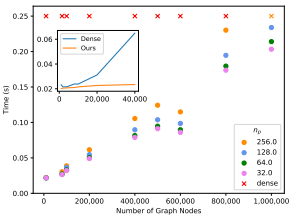


Figure 3: **Complexity analysis.** The results under different numbers of graph nodes are compared. Marker `X` denotes out-of-memory. `Dense` denotes dense pairwise relation modeling.

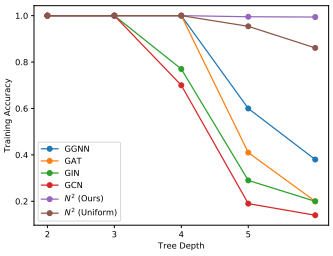
5.3 Model Analysis

5.3.1 Complexity Analysis

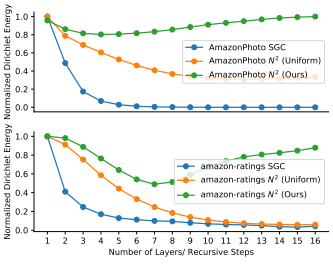
Our proposed dynamic message passing enables graph nodes to access each other without dense pairwise modeling. As described in Sec. 3.4, the space complexity of our global message passing is $O(knn_p)$. We further conduct an empirical analysis of $\mathbf{N^2}$ on computational complexity. As presented in Fig. 3, the computational time exhibits linear scalability for the number of graph nodes n, while being insensitive to the number of pseudo nodes n_p. In comparison to dense pairwise modeling, we substitute the global message passing in $\mathbf{N^2}$ with a dense attention scheme. The resulting growth rate of computational time for $\mathbf{N^2}$ is significantly lower than that of the dense method. Moreover, the dense method fails to extend to large-scale graphs, whereas $\mathbf{N^2}$ only encounters out-of-memory under the extreme situation of $n_p = 256$ and $n = 1\text{M}$.

5.3.2 Effectiveness of $\mathbf{N^2}$

Tackling over-squashing. $\mathbf{N^2}$ employs pseudo nodes in the common state space to achieve global message passing with linear complexity. These pseudo nodes also benefit $\mathbf{N^2}$ in tackling over-squashing from two aspects, *i.e.*, avoiding forming new bottlenecks on pseudo nodes and detouring from the bottlenecks on input graphs. Specifically, our pseudo nodes avoid becoming new bottlenecks by employing dynamic connections. Compared to the uniform connection, dynamic connection learns the specific edge weights for each pseudo node to different graph nodes. During global message passing, these weights can eliminate messages from certain graph nodes while preserving the others. As a result, the messages will not be squashed equally into the pseudo nodes, preventing the nodes from becoming new information bottlenecks. Regarding the bottlenecks on input graphs, the pseudo nodes produce two-hop message highways for message passing, detouring messages from these bottlenecks. To evaluate $\mathbf{N^2}$ in tackling over-squashing, we conduct experiments on Trees Match [2] with training accuracy results presented in Fig. 4(a). $\mathbf{N^2}$ maintains the fitting ability across different tree depths, demonstrating its ability to counteract over-squashing. Fig. 4(a) also verifies the superiority of the dynamic connection in tackling over-squashing, where $\mathbf{N^2}$ with the uniform connection shows a decline in accuracy as the depth increases.



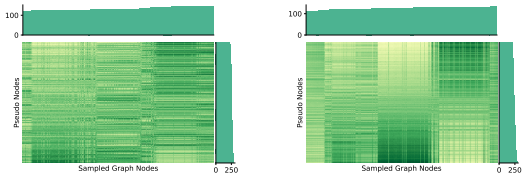
(a) Over-squashing



(b) Over-smoothing

Figure 4: **Effectiveness study.**

https://doi.org/10.52202/079017-2572



(a) $\mathbf{E}^{\mathrm{np}}$ (b) $\mathbf{E}^{\mathrm{pn}\top}$

Figure 6: **Message passing analysis.** The proximities between sampled graph nodes and pseudo nodes are depicted in the center of each sub-figure, where darker green indicates higher proximity and brighter green indicates lower proximity. The distribution on the top/right of each sub-figure denotes the sum of proximity for each graph/pseudo node. Graph/pseudo nodes are ranked based on the sum of proximity.

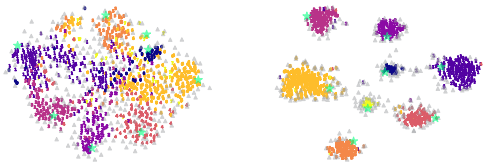
Figure 7: **Ablation on the modules of $\mathbf{N^2}$ (measured by accuracy: %).** PA., L., and G. denote pseudo-node adaptation, local message passing, and global message passing. Att. and Prx. denote the attention and proximity measurement.

		Amz -ratings	Amz Photo	RPO -TEINS
w/o.	PA.	48.20$_{\pm 0.28}$	95.10$_{\pm 0.72}$	75.76$_{\pm 2.33}$
	L.	48.69$_{\pm 0.67}$	94.84$_{\pm 0.59}$	75.77$_{\pm 2.17}$
	G.	50.16$_{\pm 0.54}$	94.58$_{\pm 0.40}$	75.54$_{\pm 1.63}$
Full with Att.		49.18$_{\pm 0.47}$	95.02$_{\pm 0.20}$	73.68$_{\pm 1.87}$
Full with Prx.		**50.25**$_{\pm 0.53}$	**95.75**$_{\pm 0.34}$	**77.53**$_{\pm 1.78}$

Tackling over-smoothing. The other issue encountered when message passing relies heavily on input topology is over-smoothing [46]. In $\mathbf{N^2}$, pseudo nodes are adopted as message pathway alternates, decoupling the message-passing process from input topology. During global message passing, our dynamic connection empowers the connected graph nodes on the input graphs to receive different outputs and avoid becoming too similar to each other. For local message passing, learning node displacements in $\mathbf{N^2}$ allows the combination of local message-passing outputs with layer inputs and global message-passing outputs. Although stacking multi-layer local message passing leads to over-smoothing, the layer inputs and global outputs can maintain the high-frequency signals. To evaluate whether $\mathbf{N^2}$ can alleviate the over-smoothing issue, we explore the changes in the Dirichlet energy [8] as the number of recursive steps increases. Fig. 4(b) compares among $\mathbf{N^2}$, $\mathbf{N^2}$ with the uniform connection, and SGC [66] on AmazonPhoto and amazon-ratings. The reported values are normalized to [0, 1] by dividing the maximum energy values. Results show that $\mathbf{N^2}$ with uniform connections encounters over-smoothing as the number of recursive steps increases, while $\mathbf{N^2}$ with dynamic connections can maintain the Dirichlet energy and alleviate over-smoothing.

5.3.3 Distribution of Embedded Nodes

We visualize the distribution of all the embedded nodes during training. Results on AmazonPhoto at the first recursive step are depicted in Fig. 5. For results through recursion, please refer to Appendix D.2. We can see that as the graph nodes are clustered with different class nodes, pseudo nodes are also split into several groups. Each pseudo-node group is attracted toward a distinct graph node cluster. This indicates that pseudo nodes will attend a particular graph node group, serving as the global message pathways to other groups. As a result, each pseudo node assumes a balanced fraction of the global message load, mitigating the risk of becoming bottlenecks for message passing.



(a) Epoch=20 (b) Epoch=500

Figure 5: **Distribution of embedded nodes.** The t-sne [61] results under different training epochs are compared. **0**, **1**, **2**, **3**, **4**, **5** and **6** denote graph nodes with different labels. △ denotes pseudo nodes. ⋆ denotes class nodes.

To further analyze the message passing between graph nodes and pseudo nodes, we visualize the proximity matrix $\mathbf{E}^{\mathrm{pn}}$ and $\mathbf{E}^{\mathrm{np}}$ for pseudo-node adaptation (Eq. 8) on AmazonPhoto in Fig. 6. For more proximity visualizations, please refer to Appendix D.3. 1000 graph nodes are sampled randomly and ranked based on their intro-/outre-proximity summation. The intro-/outre-proximity summation indicates the message load a graph node collects or diffuses during the message passing. As depicted in Fig. 6, both graph nodes and pseudo nodes assume a balanced message load. Each pseudo node has various proximity values toward individual graph nodes, different from the uniform pathways. All these properties empower efficient global message passing on graphs.

https://doi.org/10.52202/079017-2572

5.3.4 Ablation Study

Pseudo node. We conduct ablation studies on the number of pseudo nodes n_p in Fig. 8. To present the results in a comparable form within the same figure, the accuracy values have been normalized by subtracting the minimum value. As n_p increases, $\mathbf{N^2}$ gets improvement in accuracy for all three datasets. However, the accuracy exhibits degradation on PROTEINS when n_p reaches 256. This is because n_p exceeds the optimization capacity of $\mathbf{N^2}$. A similar degradation also happens on amazon-ratings. In practice, the optimal value of n_p is around 16 to 32 for graph classification while reaching 128 to 300 for node classification. We also conduct ablation studies on the engagement of pseudo nodes in Appendix D.7, where $\mathbf{N^2}$ with pseudo nodes outperforms dense message passing.

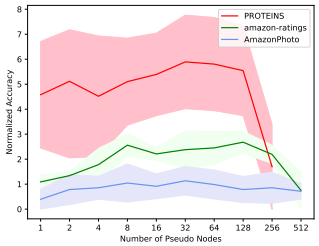


Figure 8: **Ablation on the number of pseudo nodes (n_p).**

Recurrent layer. By employing the same recurrent layer through recursive steps, $\mathbf{N^2}$ attains comparable performance to baseline models with substantially fewer parameters. To analyze the efficacy of parameter sharing, we conduct ablation studies on the number of recurrent layers, comparing $\mathbf{N^2}$ with shared parameters against $\mathbf{N^2}$ with multiple recurrent layers in Fig. 9. For simplicity, we denote the number of recurrent layers as L_p, where $\mathbf{N^2}$ with multiple recurrent layers has L_p equals to the number of recursive steps L and $\mathbf{N^2}$ with shared parameters has $L_p = 1$. Given the same L, $\mathbf{N^2}$ with $L_p = 1$ achieves matching performance with $L_p = L$. This indicates that shared parameters are sufficient in modeling convergent dynamics of the embedded nodes in the state space (Fig. 5). However, $\mathbf{N^2}$ achieves better performance with $L_p = L$ when L surpasses 8 on amazon-ratings and AmazonPhoto. In further empirical analysis, we find that the embedded nodes in $\mathbf{N^2}$ with $L_p = 1$ tend to maintain current dynamics through recursive steps and thus become less effective in precise position optimization. Please refer to Appendix D.4 for detailed analysis. A step-dependent parameter may further improve the performance of $\mathbf{N^2}$ with $L_p = 1$. We will keep exploring it in future work.

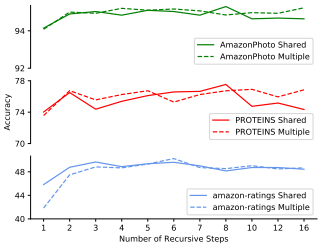


Figure 9: **Comparison between single shared recurrent layer and multiple recurrent layers.**

Relation measurement. We compare our proximity measurement with attention [62] in Tab. 7. Results show that $\mathbf{N^2}$ with proximity achieves better performance on all three benchmarks. We ascribe this to the flexible range of the proximity values. In contrast, normalized attention that ranges in $[0, 1]$ can yield equally small weights and attenuate the messages, especially when the optimal assignment involves a large number of graph nodes to the same pseudo node.

$\mathbf{N^2}$ modules. In Tab. 7, $\mathbf{N^2}$ with all three modules achieves superior performance across all the benchmarks. In comparison among the ablated $\mathbf{N^2}$, removing global message passing leads to a larger degradation in graph classification accuracy. Conversely, node classification exhibits greater sensitivity to the removal of local message passing. Moreover, pseudo-node adaptation is required by $\mathbf{N^2}$ on all three benchmarks. This indicates that adapting the randomly initialized distribution of pseudo nodes enables better interactions with graph nodes.

6 Conclusion

In this paper, we presented a dynamic message-passing method on graphs. Both graph nodes and pseudo nodes are embedded in a common state space with measurable relations between them. The measured relations serve as dynamic pathways between the embedded nodes, empowering flexible message passing. Associating pseudo nodes to input graphs with measured relations, graph nodes can communicate with each other intermediately through pseudo nodes under linear complexity. Based on the proposed dynamic message passing, we further developed a GNN model named $\mathbf{N^2}$ for graph and node classification tasks. Experimental results show that $\mathbf{N^2}$ achieves superior performance over competitive baseline models. For limitations discussion, please refer to Appendix F.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Key R&D Program of China under Grant 2023YFC2508704, in part by the National Natural Science Foundation of China: 62236008, U21B2038, and 61931008, and in part by the Fundamental Research Funds for the Central Universities. The authors would like to thank Wanxing Chang and the anonymous reviewers for their helpful comments and suggestions that improved this manuscript.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *International Conference on Machine Learning*, pages 21–29, Long Beach, USA, 2019. PMLR.

[2] Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*, Virtual Only, 2021.

[3] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate Learning of Graph Representations with Graph Multiset Pooling. In *International Conference on Learning Representations*, virtual, 2022.

[4] Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

[5] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond Low-frequency Information in Graph Convolutional Networks. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, virtual, 2021.

[6] Xavier Bresson and Thomas Laurent. Residual Gated Graph ConvNets, 2018.

[7] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the Connection Between MPNN and Graph Transformer. In *Proceedings of the 40th International Conference on Machine Learning*, pages 3408–3430. PMLR, 2023.

[8] Chen Cai and Yusu Wang. A Note on Over-Smoothing for Graph Neural Networks. *arXiv:2006.13318 [cs, stat]*, 2020.

[9] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *International Conference for Learning Representations*, 2023.

[10] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and Deep Graph Convolutional Networks. In *International Conference on Machine Learning*, pages 1725–1735, virtual, 2020. PMLR.

[11] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*, virtual, 2022.

[12] Andreea Deac, Marc Lackenby, and Petar Veličković. Expander Graph Propagation. In *Learning on Graphs Conference*, volume 198, page 38. PMLR, 2022.

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *arXiv:1606.09375 [cs, stat]*, 2017.

[14] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio', and Michael Bronstein. On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology. In *International Conference on Machine Learning*, 2023.

[15] Francesco Di Giovanni, T. Konstantin Rusch, Michael Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Veličković. How does over-squashing affect the power of GNNs? *Transactions on Machine Learning Research*, 2023.

[16] Cameron Diao and Ricky Loynd. Relational Attention: Generalizing Transformers for Graph-Structured Tasks. In *International Conference on Learning Representations*, 2023.

[17] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long Range Graph Benchmark. In *Advances in Neural Information Processing Systems*, volume 35, pages 22326–22340, 2022.

 https://doi.org/10.52202/079017-2572

[18] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *International Conference on Learning Representations Workshop on Graphs and Manifolds*, 2019.

[19] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning Discrete Structures for Graph Neural Networks. In *International Conference on Machine Learning*, pages 1972–1982. PMLR, 2019.

[20] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *International Conference on Machine Learning*, pages 2083–2092, Long Beach, California, USA, 2019. PMLR.

[21] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2018.

[22] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *International Conference on Machine Learning*, volume 70, pages 1263–1272, Sydney, Australia, 2017. PMLR.

[23] Albert Gu, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2021.

[24] Benjamin Gutteridge, Xiaowen Dong, Michael M. Bronstein, and Francesco Di Giovanni. DRew: Dynamically Rewired Message Passing with Delay. In *International Conference on Machine Learning*, pages 12252–12267. PMLR, 2023.

[25] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *International Conference on Neural Information Processing Systems*, pages 1025–1035, Red Hook, USA, 2017. Curran Associates Inc.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[27] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, volume 33, pages 22118–22133. Curran Associates, Inc., 2020.

[28] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*, 2019.

[29] Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. Going Deeper into Permutation-Sensitive Graph Neural Networks. In *International Conference on Machine Learning*, pages 9377–9409, Baltimore, Maryland, USA, 2022. PMLR.

[30] EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. Revisiting Virtual Nodes in Graph Neural Networks for Link Prediction. 2021.

[31] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. Differentiable Graph Module (DGM) for Graph Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2023.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference for Learning Representations*, San Diego, USA, 2015.

[33] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, Toulon, France, 2017.

[34] Johannes Klicpera, Stefan Weiß enberger, and Stephan Günnemann. Diffusion Improves Graph Learning. In *International Conference on Advances in Neural Information Processing Systems*, volume 32, pages 13333–13345, Vancouver, Canada, 2019. Curran Associates, Inc.

[35] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking Graph Transformers with Spectral Attention. In *Advances in Neural Information Processing Systems*, volume 34, pages 21618–21629. Curran Associates, Inc., 2021.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[37] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. In *International Conference on Machine Learning*, pages 3734–3743, Long Beach, California, USA, 2019. PMLR.

[38] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014.

[39] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018.

[40] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding Global Homophily in Graph Neural Networks When Meeting Heterophily. In *International Conference on Machine Learning*, volume 162, pages 13242–13256, Baltimore, Maryland, USA, 2022. PMLR.

[41] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In *Advances in Neural Information Processing Systems*, volume 34, pages 20887–20902. Curran Associates, Inc., 2021.

[42] Xin Liu, Jiayang Cheng, Yangqiu Song, and Xin Jiang. Boosting Graph Structure Learning with Dummy Nodes. In *International Conference on Machine Learning*, pages 13704–13716. PMLR, 2022.

[43] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. GraphiT: Encoding Graph Structure in Transformers. *arXiv preprint*.

[44] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *International Conference on Machine Learning Workshop on Graph Representation Learning and Beyond*, Virtual Only, 2020. arXiv.

[45] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, volume 48, pages 2014–2023, New York, NY, USA, 2016. PMLR.

[46] Kenta Oono and Taiji Suzuki. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *International Conference for Learning Representations*, 2021.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *International Conference on Neural Information Processing Systems*, pages 8026–8037, Red Hook, NY, USA, 2019. Curran Associates Inc.

[48] Zhengqi Pei and Shuhui Wang. Dynamics-inspired Neuromorphic Visual Representation Learning. In *International Conference on Machine Learning*, pages 27521–27541. PMLR, 2023.

[49] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda, 2023.

[50] Chendi Qian, Andrei Manolache, Kareem Ahmed, Zhe Zeng, Guy Van den Broeck, Mathias Niepert, and Christopher Morris. Probabilistically Rewired Message-Passing Neural Networks. In *International Conference on Learning Representations*, 2023.

[51] Saeed Rahmani, Asiye Baghbani, Nizar Bouguila, and Zachary Patterson. Graph Neural Networks for Intelligent Transportation Systems: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(8):8846–8885, 2023.

[52] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. In *Advances in Neural Information Processing Systems*, volume 35, pages 14501–14515, New Orleans, USA, 2022.

[53] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, New York, NY, USA, 2020.

[54] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

 https://doi.org/10.52202/079017-2572

[55] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation. *arXiv:1811.05868 [cs, stat]*, 2019.

[56] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *International Joint Conference on Artificial Intelligence*, volume 2, pages 1548–1554, Virtual Event / Montreal, Canada, 2021. ijcai.org.

[57] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse Transformers for Graphs. In *International Conference on Machine Learning*, volume 202, Honolulu, USA, 2023. PMLR.

[58] Junshu Sun, Shuhui Wang, Xinzhe Han, Zhe Xue, and Qingming Huang. All in a row: Compressed convolution networks for graphs. In *International Conference on Machine Learning*, volume 202, pages 33061–33076, Honolulu, USA, 2023. PMLR.

[59] Yifei Sun, Haoran Deng, Yang Yang, Chunping Wang, Jiarong Xu, Renhong Huang, Linfeng Cao, Yang Wang, and Lei Chen. Beyond Homophily: Structure-aware Path Aggregation Graph Neural Network. In *International Joint Conference on Artificial Intelligence*, volume 3, pages 2233–2240, 2022.

[60] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference for Learning Representations*, 2021.

[61] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *International Conference on Neural Information Processing Systems*, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[63] Petar Veličković. Message passing all the way up. In *International Conference on Learning Representation Workshop on Geometrical and Topological Representation Learning*, 2022.

[64] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, Vancouver, Canada, 2018.

[65] Huizhao Wang, Yao Fu, Tao Yu, Linghui Hu, Weihao Jiang, and Shiliang Pu. PROSE: Graph Structure Learning via Progressive Strategy. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2337–2348, New York, NY, USA, 2023. Association for Computing Machinery.

[66] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*, pages 6861–6871, Long Beach, USA, 2019. PMLR.

[67] Junran Wu, Xueyuan Chen, Shangzhe Li, and Ke Xu. Structural entropy guided graph hierarchical pooling. In *International Conference on Machine Learning*, volume 162, pages 24017–24030, Baltimore, Maryland, USA, 2022. PMLR.

[68] Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[69] Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification. In *Advances in Neural Information Processing Systems*, 2022.

[70] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and Empowering Transformers for Large-Graph Representations. In *Conference on Neural Information Processing Systems*, 2023.

[71] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[72] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Bad for Graph Representation? In *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888. Curran Associates, Inc., 2021.

[73] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, volume 31, pages 4805–4815, Montréal, Canada, 2018. Curran Associates, Inc.

[74] Hao Yuan and Shuiwang Ji. StructPool: Structured Graph Pooling via Conditional Random Fields. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[75] Sheheryar Zaidi, Michael Schaarschmidt, James Martens, Hyunjik Kim, Yee Whye Teh, Alvaro Sanchez-Gonzalez, Peter Battaglia, Razvan Pascanu, and Jonathan Godwin. Pre-training via Denoising for Molecular Property Prediction. In *The Eleventh International Conference on Learning Representations*, 2023.

[76] Wentao Zhao, Qitian Wu, Chenxiao Yang, and Junchi Yan. GraphGLOW: Universal and Generalizable Structure Learning for Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3525–3536, New York, NY, USA, 2023. Association for Computing Machinery.

[77] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[78] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *International Conference on Neural Information Processing Systems*, pages 7793–7804, Red Hook, NY, USA, 2020. Curran Associates Inc.

[79] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Structure Learning for Robust Representations: A Survey. *arXiv:2103.03036 [cs]*, 2021.

 https://doi.org/10.52202/079017-2572

A State Space

$\mathbf{N^2}$ unifies graph nodes and pseudo nodes in a common state space. The concept of state space has been widely adopted in various machine learning methods. In reinforcement learning, the state space is typically modeled as a discrete space, encompassing all possible states of an agent. By taking discrete actions, agents make transitions between states. Through interactions with the environment, agents learn optimal policies that determine their actions at certain states, maximizing the cumulative reward over time.

In contrast to the discrete paradigm, recurrent models such as long short-term memory (LSTM) [26], and state space models (SSMs) [23] utilize continuous state representations to model sequential data. In this setting, input sequences are tokenized and consumed by the model in a successive manner. As a result, recurrent models learn to update the state embeddings recursively.

Building upon the continuous paradigm, DyN [48] further applies measurements to the continuous state space, giving rise to measurable spatial relations between state embeddings. All the input tokens can now be modeled simultaneously in the state space.

However, modeling dense pairwise relations among a large number of graph nodes becomes intractable in the graph representation learning setting, due to the quadratic complexity. To reduce the complexity, we introduce pseudo nodes to serve as proxies for pairwise relations between graph nodes. Consequently, the relations between graph nodes are decomposed into two components, *i.e.*, relations between source graph nodes and pseudo nodes, and relations between pseudo nodes and target graph nodes. Since the number of pseudo nodes is substantially smaller than the number of graph nodes, this decomposition effectively reduces the overall complexity.

B Pseudo Nodes and Pooling Nodes

In hierarchical GNNs, graph nodes are compressed into higher-level nodes through iterative graph pooling. These higher-level nodes, referred to as pooling nodes, are different from the pseudo nodes in $\mathbf{N^2}$. First, pseudo nodes and pooling nodes are learned for different objectives. Pseudo nodes optimize communication efficiency between graph nodes during global message passing. Pooling nodes capture the hierarchy in the graph structures, ensuring better structure compression. Second, pooling nodes and pseudo nodes have different relations with input graph nodes. Pooling nodes are higher-level abstractions of graph nodes and are not physically connected with them. In contrast, pseudo nodes can be regarded as learnable graph nodes. These nodes are physically connected to graph nodes as part of the graphs and directly participate in the message passing between graph nodes. Third, pseudo nodes and pooling nodes employ different communication pathways. Pooling nodes employ the coarse adjacency matrix for message passing. Pseudo nodes are free from structure-preserving constraints and learn fully connected pathways to communicate with each other.

C Details on Experiments

We have performed grid search for the hyper-parameters in Tab. S6 and the reproduced models based on validation loss. All the learnable parameters in $\mathbf{N^2}$ are optimized during training, including the weights in the linear transformations and the proximity measurement, the pseudo/class-node states. The cross-entropy loss is adopted for classification and the L1 loss for regression. No position or structural encoding methods are employed for $\mathbf{N^2}$.

C.1 Graph Classification

C.1.1 Benchmark Descriptions

We adopt six benchmarks for graph classification, including three biochemical datasets (OGB-molpcba [27], PROTEINS [44], NCI1 [44]) and three social network datasets [44] (COLLAB, IMDB-BINARY, and IMDB-MULTI).

OGB-molpcba is a large-scale molecular property prediction benchmark. Each graph is a discrete molecule, wherein nodes denote individual atoms and edges encode chemical bonds between atoms.

PROTEINS comprises 1,113 protein graphs with amino acids constituting the nodes. The associated binary classification task involves predicting protein category labels, specifically discriminating between enzymes versus non-enzymes.

NCI1 constitutes 4,110 graphs of chemical compounds assembled by the National Cancer Institute (NCI). Graph labels categorize compounds as exhibiting either positive or negative efficacy against cell lung cancer.

IMDB-BINARY and **IMDB-MULTI** constitute collaboration network graphs of movie actors and actresses, with graph labels indicating movie genres.

COLLAB constitutes scientific collaboration networks wherein each graph is an ego network for a researcher, encompassing their co-authors. Graph labels indicate the specific scientific interest corresponding to each researcher.

C.1.2 Experimental Setups

We choose (**1**) Convolutional GNNs: GCN [33], PATCHY-SAN [45], GraphSAGE [25], GIN [71], PG [29], and CoCN [58]; (**2**) GNNs with a single pseudo node from [42, 27, 7]; (**3**) Hierarchical GNNs: DiffPool [73], TopKPool [20], SAGPool [37], StructPool [74], SEP [67], and GMT [3]; (**4**) Graph transformers: Graphormer [72], SAN [35], GraphGPS [52], and GraphTrans [7] as the baselines of graph classification.

Except for OGB-molpcba, we perform 10-fold cross-validation with LIB-SVM following [71] and report average performance. Average precision is reported for OGB-molpcba while accuracy is for the others. Since COLLAB, IMDB-BINARY, and IMDB-MULTI have no graph node features, we use the one-hot encoding of node degrees as node features following [71].

For experiments on all benchmarks, the learning rate is set to 1×10^{-3}. We adopt Adam [32] as optimizer and set weight decay as 1×10^{-6}. Early stopping regularization is employed, where we stop the training if there is no further reduction in the validation loss during 300 epochs. The maximum epoch number is set to 1,000. The batch size is set to 1,024 on OGB-molpcba, 256 on PROTEINS, NCI1, IMDB-BINARY, IMDB-MULTI, and COLLAB. The detailed hyper-parameter settings on all benchmarks are reported in Tab. S6.

C.2 Node Classification

C.2.1 Benchmark Descriptions

For node classification, we conduct experiments on (**1**) six middle-scale benchmarks: homophilic graphs [55] (AmazonPhoto, AmazonComputers, CoauthorCS, and CoauthorPhysics), heterophilic graphs [49] (questions, amazon-ratings, tolokers, and minesweeper); (**2**) four large-scale benchmarks: homophilic graphs [27] (OGB-arXiv, OGB-proteins), heterophilic graphs [41] (arXiv-year, genius).

CoauthorCS and CoauthorPhysics originate from the Microsoft Academic Graph, comprising co-authorship graphs wherein nodes denote researchers and edges denote co-authorships between pairs. Node features encapsulate keyword frequencies extracted from an author's publications. Graph labels categorize the dominant research interest in computer science or physics for each author.

AmazonComputers and AmazonPhoto constitute Amazon co-purchase graphs. Nodes denote products that are available for purchase and edges denote co-purchase relation between pairs of items. Node features are encoded customer review texts corresponding to each product. Graph labels categorize the products.

Questions originates from the Yandex Q question-answering website, comprising user activity graphs over a one-year interval (September 2021 to August 2022). Nodes represent users with interest in "medicine". Edges denote one user answered another user's posted question. The associated binary graph classification task involves predicting which users remained active on the website without account deletion or blocking. Node features are derived by averaging FastText word embedding vectors corresponding to user profile descriptions.

Amazon-ratings utilizes the Amazon product co-purchasing network metadata sourced from the SNAP Datasets [38]. Nodes are products with edges encoding frequent co-purchase relations between item pairs. The associated task involves predicting the average reviewer rating for each product,

 https://doi.org/10.52202/079017-2572

Table S6: **Hyper-parameter setups for $\mathbf{N^2}$.**

	#Recursive Steps (L)	Hidden Dim.	State Space Dim.	#Units (k)	#Pseudo Nodes (n_p)	Dropout
Genius	3	128	64	8	256	0.1
arXiv-year	6	128	64	8	300	0.1
OGB-arXiv	5	128	64	8	320	0.3
OGB-proteins	3	128	64	8	256	0.3
questions	6	128	64	8	256	0.1
amazon-ratings	8	64	64	8	256	0.3
tolokers	6	128	64	8	256	0.1
minesweeper	7	64	64	8	128	0.3
CoauthorCS	3	128	64	8	256	0.4
CoauthorPhysics	3	128	64	8	256	0.5
AmazonPhoto	5	128	64	8	256	0.1
AmazonComputers	3	128	64	8	256	0.3
PROTEINS	8	128	64	8	32	0.1
NCI1	6	128	128	8	32	0.1
COLLAB	6	128	64	8	32	0.1
IMDB-BINARY	6	128	64	8	32	0.1
IMDB-MULTI	6	128	64	8	8	0.2
OGB-molpcba	6	128	64	8	32	0.3

which is grouped into five ordinal rating classes. Node features are derived by averaging FastText embedding representations corresponding to each product's description text.

Tolokers encapsulates crowdsourcing participation data sourced from the Toloka platform. Nodes denote contributors, referred to as "tolokers", involved in at least one out of 13 selected projects. Edges link toloker pairs that have completed the same tasks. The associated binary classification is to predict which tolokers have been banned from projects. Node features are profile attributes and task performance statistics of each toloker.

Minesweeper is a synthetic 100x100 grid network. Nodes denote grid cells. 20% of nodes are randomly designated as mines. The associated prediction task is to classify which nodes are mine or not. For all nodes, input features are initialized as one-hot vectors encoding counts of neighboring mines. The initialized features of 50% of randomly selected nodes are then reset to unknown values. These nodes are indicated by an additional binary indicator.

OGB-arXiv is a citation network of Computer Science papers on arXiv. Nodes represent individual articles whereas directed edges denote one paper citing another. Node features are derived by averaging 128-dimensional word embeddings corresponding to the title and abstract of each publication. The associated multi-class classification is to predict the primary category for arXiv articles across 40 classes.

OGB-proteins is an undirected, weighted graph of proteins. Nodes denote proteins while edges denote biologically meaningful associations between proteins. Edge features represent confidence scores for each association type. Node features are one-hot vectors denoting the species of each protein. The associated multi-label classification is to predict the presence of 112 potential protein functions, formulated as a binary prediction task for each label.

Genius is an online social network. Each node represents a user. The associated binary classification task is to predict whether the accounts are marked or not.

arXiv-year is a citation network from arXiv. Each node represents a research publication. The associated classification task is to predict the publication time of each node.

Table S7: **Effectiveness study on peptides-struct (measured by mean absolute error).**

	PEPTIDES-STRUCT ↓
GCNII	34.71
GCN	34.96
GINE	35.47
GATEDGCN	33.57
SAN	25.45
PATHNN	25.45
DREW-GCN	25.36
EXPHORMER	24.81
$\mathbf{N^2}$ **(OURS)**	25.12

C.2.2 Experimental Setups

For baseline models, we consider (**1**) Convolutional GNNs: GCN [33], GAT [64], APPNP [21], SGC [66], H2GCN [78], FAGCN [5], LINKX [41], GPRGNN [11], and GloGNN [40]; (**2**) Graph transformers: GT [56], Graphormer [72], SAN [35], GraphGPS [52], Nodeformer [69], NAGphormer [9], SGFormer [70] and Exphormer [57] based on pseudo node.

ROC-AUC is reported for questions, tolokers, and minesweeper while accuracy is for the others. We apply 60%/20%/20% train/val/test random splits for Amazon and Coauthor benchmarks and follow the standard splits as the original papers for the rest of the benchmarks. We reproduce the results of Exphormer [57] on (genius, arXiv-year, OGN-Proteins, questions, amazon-ratings, tolokers, minesweeper), Nodeformer [69] and SGFormer [70] on (genius and arXiv-year) with their released codes for a fair comparison.

For experiments on all benchmarks, the learning rate is set to 1×10^{-3}. We adopt Adam [32] as optimizer and set weight decay as 1×10^{-6}. Early stopping regularization is employed, where we stop the training if there is no further reduction in the validation loss during 300 epochs. The maximum epoch number is set to 1,000. The detailed hyper-parameter settings on all benchmarks are reported in Tab. S6.

D Additional Experimental Results

D.1 Effectiveness on Tackling Over-squashing

We further benchmark $\mathbf{N^2}$ on Peptides-struct [17], a graph regression benchmark involving long-range interactions. Baselines include traditional GNNs: GCN [33], GCNII [10], GINE [71, 28], and GatedGCN [6]; methods aiming to capture long-range features: SAN [35], PathNN [59], Drew-GCN [24], and Exphormer [57]. As presented in Tab. S7, we can see that $\mathbf{N^2}$ achieves competitive performance with Exphormer and surpasses the rest of baselines. This indicates the effectiveness of $\mathbf{N^2}$ to encounter over-squashing.

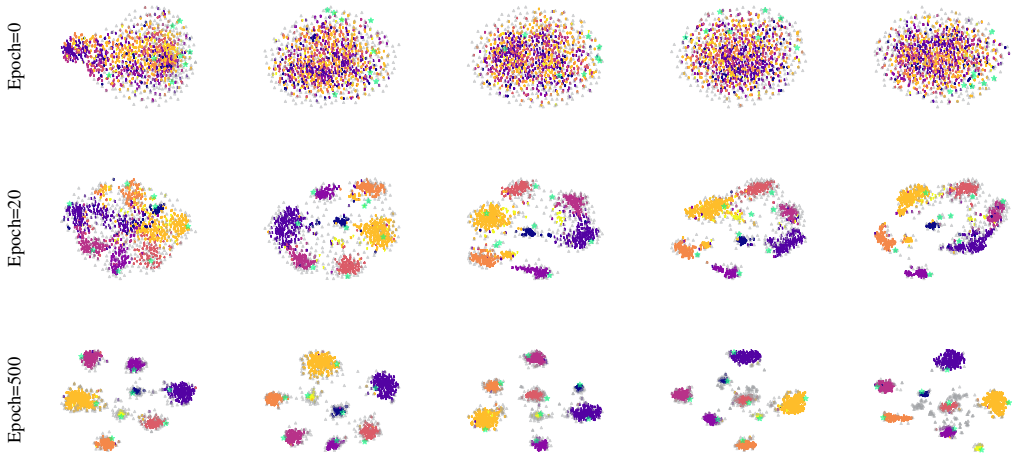
D.2 Distribution of Embedded Nodes

The distribution of the embedded nodes through training is visualized in Fig. S10. We can see that pseudo nodes and graph nodes adjust their relative position actively in the state space. For the same recursive step through training, pseudo nodes are split into several groups. Each group is attracted toward a distinct graph node cluster. For the same epoch, the relative positions between graph node clusters and the attracted pseudo nodes also evolve through recursive steps. This indicates that $\mathbf{N^2}$ optimizes the proximity between graph nodes and pseudo nodes recursively, constructing dynamic message-passing pathways.

D.3 Proximity for Message Passing

To further analyze the message passing between graph nodes and pseudo nodes, we visualize their corresponding proximity on AmazonPhoto. In Fig. S11, $1,000$ graph nodes are sampled randomly.

 https://doi.org/10.52202/079017-2572



(a) $l = 0$ (b) $l = 1$ (c) $l = 2$ (d) $l = 3$ (e) $l = 4$

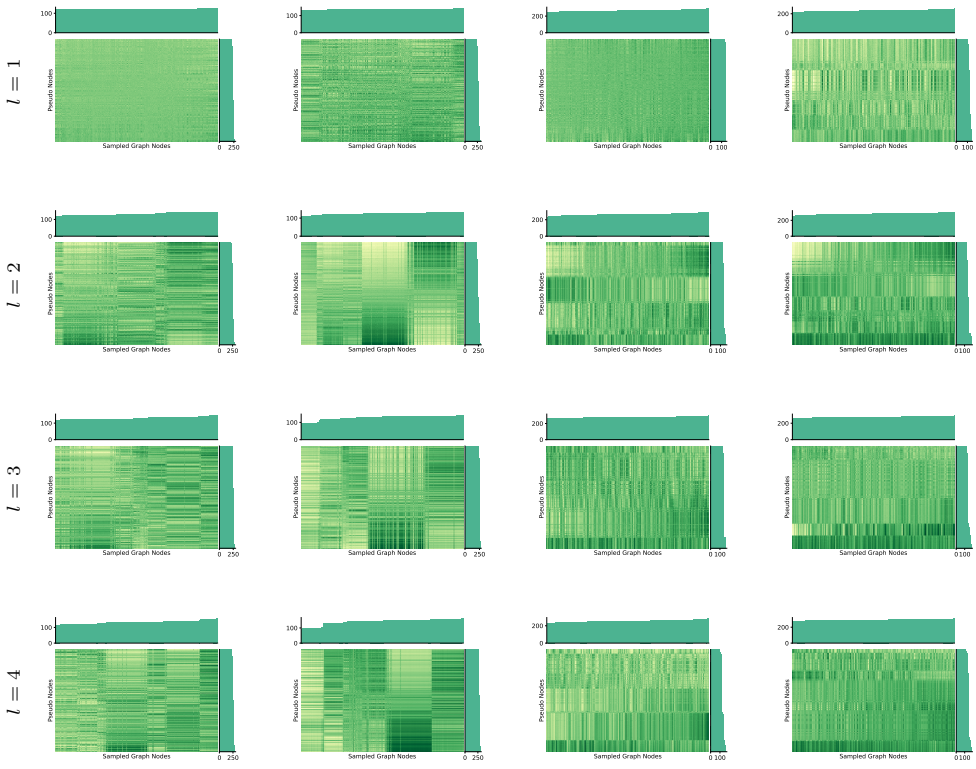
Figure S10: **Full results on the distribution of embedded nodes.** The t-sne [61] results under different recursive steps across training are compared. Each row is from the same epoch. Each column depicts results from the same recursive step (l-th step). Input graph nodes with different labels are depicted as: **0**, **1**, **2**, **3**, **4**, **5**, **6**. Pseudo nodes are depicted as △. Class nodes are depicted as ⋆.

Both graph nodes and pseudo nodes are ranked based on their intro-/outre-proximity summation. The intro-/outre-proximity summation indicates the message load a graph node takes or emits during the message passing. From Fig. S11, we can see that the intro-/outre-proximity is distributed evenly across different graph nodes and pseudo nodes. This indicates that both graph nodes and pseudo nodes assume a balanced message load. For each proximity matrix, the proximity value varies between different pairs of graph nodes and pseudo nodes, thus constructing dynamic pathways instead of uniform pathways.

D.4 Displacements Comparison between Single Shared Recurrent Layer and Multiple Recurrent Layers

To understand the difference in multi-step performance, where $\mathbf{N^2}$ with multiple recurrent layers ($L_p = L$) achieves better performance against $\mathbf{N^2}$ with shared parameters ($L_p = 1$), we further analyze the displacements of the embedded nodes on AmazonPhoto and amazon-ratings. Fig. S12 depicts the comparison results. Four displacement types are presented in the figure, including displacements of pseudo nodes in pseudo-node adaptation $\Delta\hat{\mathbf{R}}^{(l)}$ (Eq. 8) and global message passing $\Delta\mathbf{R}^{(l)}$ (Eq. 10), displacements of graph nodes in local message passing $\texttt{NL}(\mathbf{M}^{\texttt{local}(l)})$ (Eq. 9) and global message passing $\texttt{NL}(\mathbf{M}^{\texttt{glob}(l)})$ (Eq. 10). We take the Frobenius norm of each displacement matrix and divide it by the larger results between $\mathbf{N^2}$ with $L_p = 1$ and $\mathbf{N^2}$ with $L_p = L$ across recursive steps, *e.g.*, $\texttt{max}\left(\{\Delta\mathbf{R}^{(l)}_{L_p=1}, \Delta\mathbf{R}^{(l)}_{L_p=L}, l \in [1, L]\}\right)$.

From Fig. S12, we can see the consistency in the shape of the displacement curve between amazon-ratings and AmazonPhoto. The displacement curves of $\mathbf{N^2}$ with $L_p = 1$ are smooth, whereas $\mathbf{N^2}$ with $L_p = L$ demonstrates fluctuation. We attribute this difference to the distinct inertia characteristics exhibited by the embedded nodes. For the embedded nodes in $\mathbf{N^2}$ with $L_p = 1$, they generally possess greater inertia, and thus tend to maintain current dynamics through recursive steps. In contrast, the embedded nodes in $\mathbf{N^2}$ with $L_p = L$ have smaller inertia and vary their dynamics. Therefore, $\mathbf{N^2}$ with $L_p = L$ can adjust the displacements of the embedded nodes more flexibly according to different situations and gain better performance in situations requiring precise distribution optimization. A step-dependent parameter may further improve the performance of $\mathbf{N^2}$ with $L_p = 1$ on a larger number of recursive steps. We leave this for future exploration.



(a) PNode. Adapt. $\mathbf{E}^{\mathbf{np}}$ (b) PNode Adapt. $\mathbf{E}^{\mathbf{pn}\top}$ (c) Dy. MP. $\mathbf{E}^{\mathbf{np}}$ (d) Dy. MP. $\mathbf{E}^{\mathbf{pn}\top}$

Figure S11: **Full results on the message passing analysis.** The results under different recursive steps across training are compared. Each row is from the same recursive step (l-th step). PNode. Adpat. denotes the pseudo-node adaptation module in $\mathbf{N^2}$. Dy. MP. denotes the dynamic message passing in $\mathbf{N^2}$. The proximities between sampled graph nodes and pseudo nodes are depicted in the center of each sub-figure, where darker green indicates higher proximity and brighter green indicates lower proximity. Each column of the proximity matrix associates with a graph node while each row associates a pseudo node. The distribution on the top of each sub-figure denotes the sum of proximity for each graph node while the distribution on the right is for each pseudo node. The sampled graph nodes and pseudo nodes are ranked based on the sum of proximity.

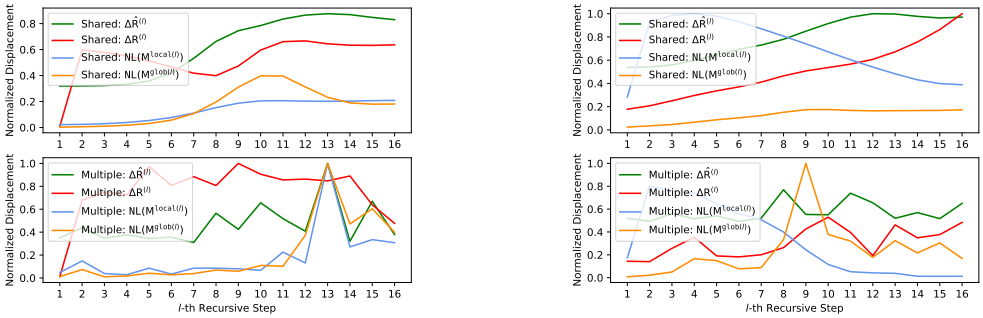
D.5 Ablation on Proximity Measurement

To approximate non-linear relations with low complexity, we employ piece-wise weighted inner products on the proximity measurement in Eq. 1. Each embedded node is divided into k pieces. Ablation studies are conducted on k with amazon-ratings, AmazonPhoto, and PROTEINS. As depicted in Fig. S13, $\mathbf{N^2}$ with multiple pieces outperforms a single piece on all three benchmarks and gains improvement with k increasing. The optimal number of pieces k is around 8.

D.6 Ablation on Messages

In $\mathbf{N^2}$, graph nodes and pseudo nodes perform message passing based on their current states in the common state space and pass on the learned messages to each other. States and messages take different roles in our proposed method. Specifically, states are the descriptive embeddings of graph nodes corresponding to specific tasks, and messages are employed for information exchange. For the message passing from graph nodes to pseudo nodes, the messages contain the features of an input graph. From pseudo nodes to graph nodes, the messages may contain the query information towards input graphs. To evaluate the necessity of distinguishing states and messages, we conduct ablation studies on messages by directly passing graph node states in message passing. The results with or

https://doi.org/10.52202/079017-2572



(a) amazon-ratings (b) AmazonPhoto

Figure S12: **Displacement comparison between $\mathbf{N^2}$ with a shared recurrent layer and $\mathbf{N^2}$ with multiple recurrent layers.**

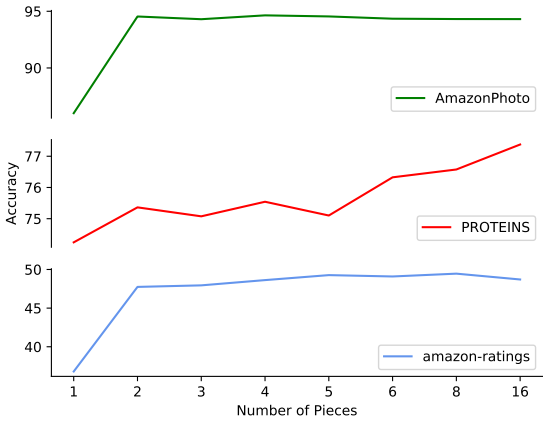


Figure S13: **Ablation studies on the number of pieces k.**

without specially distinguished messages are presented in Tab. S8. We can see that applying both states and messages achieves the best performance.

D.7 Ablation on Pseudo Nodes

In addition to investigating the number of pseudo nodes, we conduct ablation studies to examine the impact of including pseudo nodes within $\mathbf{N^2}$. For cases without pseudo nodes, dense message passing is employed. However, as dense computation is not scalable, only small-scale benchmarks are utilized for this ablation study. The results are presented in Table S9. Surprisingly, we find that $\mathbf{N^2}$ with pseudo nodes outperforms dense computation on IMDB-B, IMDB-M, and PROTEINS. This can be attributed to pseudo nodes functioning as information filters in global message passing, facilitating the removal of redundant information while extracting discriminative features from inputs.

E Intuitions for the $\mathbf{N^2}$ Implementation

We provide more intuitions for the implementation in Section 4. $\mathbf{N^2}$ embeds graph nodes and pseudo nodes into the common state space, employing a recurrent layer to parameterize the displacements of the embedded nodes. The recurrent layer includes pseudo-node adaptation and dynamic message passing. Pseudo-node adaptation employs `GlobMP` to generate query messages toward graph nodes. Dynamic message passing then extracts graph features through `LocalMP` and refines these extracted features at the pseudo-node level with `GlobMP`.

Table S8: **Ablation studies on messages.**

	W. MESSAGES	W/O. MESSAGES	W/O.-W.
QUESTIONS	**78.07**	76.01	-2.06
AMAZON-RATINGS	**50.25**	47.79	-2.46
TOLOKERS	**86.25**	85.05	-1.20
MINESWEEPER	**93.97**	92.38	-1.59
COAUTHORCS	**94.44**	92.39	-2.05
COAUTHORPHYSICS	**97.56**	96.27	-1.29
AMAZONPHOTO	**95.75**	93.80	-1.95
AMAZONCOMPUTERS	**92.51**	90.73	-1.78

Table S9: **Comparison between dense and pseudo-node-based message passing.**

	DENSE	$\mathbf{N^2}$
IMDB-B	77.94	**79.95**
IMDB-M	56.41	**57.31**
PROTEINS	76.94	**77.53**

As described by the pseudo-node adaptation in Eq. 8, graph nodes first diffuse messages to pseudo nodes. Based on the information learned from these messages, pseudo nodes generate displacements to adjust their own representations, enabling better interactions with graph nodes. Subsequently, graph nodes collect responding messages from pseudo nodes, which serve as the query signals toward specific information of the input graphs.

During the dynamic message passing, Eq. 9 presents the local message passing on input graphs. Graph nodes collect query messages from pseudo nodes, and process them together with their own generated messages and states through the `LocalMP` function. As a result, graph nodes learn the features of the input graphs and generate feedback messages to pseudo nodes accordingly. The generated messages containing the features of input graphs can also be leveraged to determine the displacements of graph nodes.

Eq. 10 presents how graph nodes perform global message passing intermediately through pseudo nodes. Pseudo nodes receive the feedback messages and again generate their displacements. As feedback messages are aggregated at the pseudo-nodes level, more global information is incorporated, guiding the movements of graph nodes and their message update.

F Limitations

$\mathbf{N^2}$ employs shared parameters to update the distribution of pseudo nodes and graph nodes recursively. We only studied $\mathbf{N^2}$ with a simple update mechanism. As discussed in the model analysis section, $\mathbf{N^2}$ encounters performance degradation when the number of recursive steps increases. In addition, the learnable pseudo nodes can be regarded as parameters in GNNs, classified in line with neurons of GNNs. Under this interpretation of pseudo nodes, only a subset of neurons from GNNs are embedded in the common state space within our framework. Comprehensively bridging all neurons and graph nodes remains an open research direction.

G Societal Impact

This paper proposed a novel method for constructing dynamic message-passing pathways by bridging graph nodes and pseudo nodes in a common state space. Our goal is to advance the field of graph representation learning. The proposed method remains independent of specific downstream applications. As graph data are ubiquitous in the real world, there are many potential applications of our work, including computational biology [75], intelligent transportation [51], and algorithmic reasoning [16]. Regarding ethical considerations, we do not presently foresee evident issues or potential for adverse societal impacts.

https://doi.org/10.52202/079017-2572

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This paper discusses the limitations of the work in Appendix F.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not provide theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Detailed experimental setups are provided in Appendix C. Code is provided at `https://github.com/sunjss/N2`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

https://doi.org/10.52202/079017-2572

Answer: [Yes]

Justification: Code with instructions for reproduction is provided at `https://github.com/sunjss/N2`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed experimental setups are provided in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Please refer to Tab. 1-5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to the beginning of Sec. 5 and Fig. 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please refer to Appendix G.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-2572

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Please refer to Appendix C.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Please refer to the supplementary materials and our open source code at `https://github.com/sunjss/N2`.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-2572