Multi-Track Message Passing: Tackling Oversmoothing and Oversquashing in Graph Learning via Preventing Heterophily Mixing

Hongbin Pei [1] **Yu Li** [1] **Huiqi Deng** [2] **Jingxin Hai** [1]
Pinghui Wang [1] **Jie Ma** [1] **Jing Tao** [1] **Yuheng Xiong** [1] **Xiaohong Guan** [1]

Abstract

The advancement toward deeper graph neural networks is currently obscured by two inherent issues in message passing, *oversmoothing* and *oversquashing*. We identify the root cause of these issues as information loss due to *heterophily mixing* in aggregation, where messages of diverse category semantics are mixed. We propose a novel multi-track graph convolutional network to address oversmoothing and oversquashing effectively. Our basic idea is intuitive: if messages are separated and independently propagated according to their category semantics, heterophilic mixing can be prevented. Consequently, we present a novel multi-track message passing scheme capable of preventing heterophilic mixing, enhancing long-distance information flow, and improving separation condition. Empirical validations show that our model achieved state-of-the-art performance on several graph datasets and effectively tackled oversmoothing and oversquashing, setting a new benchmark of 86.4% accuracy on Cora.[1]

1. Introduction

Graph neural networks (GNNs) have proven to be powerful for learning from graph-structured data (Goller & Kuchler, 1996; Scarselli et al., 2008; Kipf & Welling, 2017; Battaglia et al., 2018), showing success in various applications such as recommender systems (Wang et al., 2021), computational chemistry (Pei et al., 2024a), and physical simulation (Zhao et al., 2022). These successes can be largely attributed to *Message Passing* mechanism (Gilmer et al., 2017).

[1]MOE KLINNS Lab, Xi'an Jiaotong University, China [2]Shanghai Jiao Tong University, China. Correspondence to: Jing Tao <peihongbin@xjtu.edu.cn, liyu1998@stu.xjtu.edu.cn, jtao@xjtu.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]Code of experiments in this paper: https://github.com/XJTU-Graph-Intelligence-Lab/mtgcn

However, the advancement toward deeper GNNs is currently obscured by two inherent issues in the message passing framework: *oversmoothing* (Li et al., 2018) and *oversquashing* (Alon & Yahav, 2021). Although considerable progress has been made in tackling oversmoothing (Zhou et al., 2020; 2021b; Giovanni et al., 2023; Wang et al., 2023) and in tackling oversquashing (Topping et al., 2022; Abboud et al., 2022; Black et al., 2023; Gutteridge et al., 2023), these two issues remain insufficiently resolved to date.

The oversmoothing refers to the issue that as GNNs become deeper and undergo multiple rounds of message passing, all node representations tend to converge towards the same constant value, resulting in these nodes becoming indistinguishable, as illustrated in Fig. 1 A1. The oversquashing is the issue that abundant messages from distant nodes are "squashed" into fixed-sized representation vectors. This issue is particularly severe on heterophilic graphs (Pei et al., 2020), and on graphs with long-distant dependences (Di Giovanni et al., 2023), as illustrated in Fig. 1 B1.

In this paper, we argue that the issues of oversmoothing and oversquashing are both rooted in information loss resulting from **heterophily mixing** in aggregation of message passing, *i.e., the mixture of messages with different semantics (e.g., categories information)*. Specifically, oversmoothing happens when nodes belonging to different categories mutually mix their features, leading to a loss of their unique and discriminative information. Oversquashing occurs when features from distant nodes within an exponentially large receptive field, containing a great diversity of category semantics, are heterophily mixed into fixed-sized representation vectors with a limited capacity for information. Consequently, information is inevitably lost in this mixing.

Leveraging this understanding, we propose a novel **M**ulti-**T**rack **G**raph **C**onvolutional **N**etwork (MTGCN) to effectively address the oversmoothing and oversquashing issues. The basic idea behind MTGCN is intuitive: *If messages are separated and independently propagated according to their category semantics, heterophilic mixing can be prevented.* To realize the idea, we propose a novel concept, message tracks, which are a set of isomorphic graphs sharing the same topology with graph. Each track is dedicated

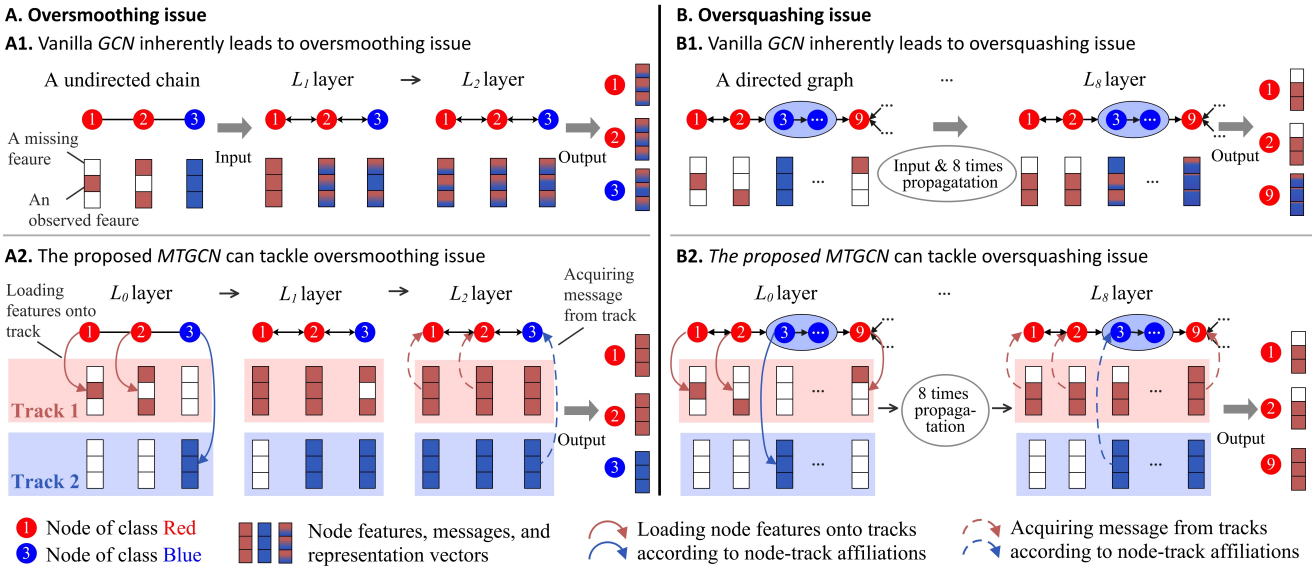

Figure 1. Illustration of our proposed MTGCN on graphs, where vanilla GCN fails due to oversmoothing and oversquashing. (A1). Oversmoothing: after two rounds of message propagation, node representations become indistinguishable. (B1). Oversquashing: node ⑨ cannot receive complete messages from distant nodes ① and ②, leading to a loss of usful information. In (A2) and (B2), MTGCN tackles oversmoothing and oversquashing by independently propagating messages according to their categories.

to passing messages of a specific category semantic.

In the MTGCN, as illustrated in Fig. 1 A2 and B2, node features are firstly loaded onto corresponding tracks as messages, where nodes belonging to the same category are associated with the same track; then, messages are updated by propagating and aggregating in respective tracks over several iterations; finally, nodes acquire the updated messages in their affiliated tracks to construct their representations as output. By independently propagating messages with different category semantics, MTGCN maintains the semantic purity of messages and outputs distinctive node representation vectors. On the same graphs, vanilla GCN fails due to oversmoothing and oversquashing.

The core of MTGCN is the proposed **M**ulti-**T**rack **M**essage **P**assing (MTMP) scheme, which propagates and aggregates messages in respective tracks. The MTMP gains improvements by three aspects: *preventing the heterophily mixing*, *facilitating long-distant information flow*, and *enhancing separation condition in semi-supervised learning*. In addition, we propose an attention method to calculate node-track affiliations, which are a crucial precondition for MTMP. We design a multi-stage training pipeline for MTGCN in a self-evolutionary manner. Experimental results on several benchmark graphs show that the MTGCN achieved state-of-the-art results on node classification and effectively tackled both oversmoothing and oversquashing issues.

In summary, the contribution of this paper is three-fold: (1) We propose a novel graph model, MTGCN, to effectively tackle oversmoothing and oversquashing by preventing heterophily mixing; (2) We present the rationales of improvements achieved by the MTMP scheme from the perspectives of graph learning and semi-supervised learning; (3) We validate and analyze MTGCN via extensive comparisons with state-of-the-art methods on several benchmark graphs.

2. Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. The nodes are associated with a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$, where $|\mathcal{V}|$ denotes the number of nodes and m denotes the number of input features per node. Let $\mathbf{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ denote the adjacency matrix corresponding to the edges and $\mathbf{D}$ the diagonal degree matrix. The symmetric normalized adjacency matrix is defined as $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, where $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ represent the degree and adjacency matrices enhanced with self-loops. The normalized adjacency matrix is frequently utilized for node aggregation in Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017). A notable example is the Graph Convolutional Network (GCN) (Kipf & Welling, 2017), which achieves node aggregation by defining the graph convolution operation as $\mathbf{H}^{(\ell+1)} = \sigma((\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2})\mathbf{H}^{(l)}\mathbf{W}^{(l)})$. Here, $\mathbf{H}^{(\ell)}$ represents the hidden features of nodes at the ℓ-th layer, $\mathbf{W}^{(\ell)}$ is a layer-specific learnable weight matrix, and $\sigma(\cdot)$ denotes a nonlinear activation function, *e.g.,* ReLU.

3. Multi-track Graph Convolutional Network

We first present the concept of *message track*, which is a fundamental element of the proposed Multi-Track Graph Convolutional Network (MTGCN).

Definition 3.1. For a graph $\mathcal{G}$, *message tracks* $\mathcal{T}$ are defined as a set of isomorphic graphs mirroring the topology of $\mathcal{G}$. Each track $T \in \mathcal{T}$ is uniquely used for passing messages corresponding to a specific category of nodes, *i.e.,* $|\mathcal{T}| = n$ where n denotes the number of node categories in $\mathcal{G}$.

We outline the proposed MTGCN by its three key steps, which are also illustrated in Fig. 1A2, B2 and Fig. 3A.

Step 1: Loading. All nodes' raw features are loaded onto corresponding tracks $\mathcal{T}$ as initial messages $\mathcal{M}^{(0)}$. Nodes belonging to the same category are expected to associate with the same track, governed by a node-track affiliation matrix $\mathbf{F} \in \{0,1\}^{|\mathcal{T}|\times|\mathcal{V}|}$. Here, each entry $\mathbf{F}_{T,v}$ guides whether the features of node v are loaded onto track T.
Step 2: Multi-Track Message Passing (MTMP). The initial messages are updated by propagating and aggregating in respective tracks over L iterations.
Step 3: Acquiring. Based on the affiliations $\mathbf{F}$, nodes acquire the updated messages $\mathcal{M}^{(L)}$ in their affiliated tracks to construct their node representation $\mathbf{Z}$.

Next, we elaborate on the MTMP and the affiliations $\mathbf{F}$, respectively, which are the most important in MTGCN.

3.1. Multi-track Message Passing

In MTMP, we model messages at all nodes in all tracks as a 3rd-order tensor $\mathcal{M} \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{V}|\times d}$, where d is the dimension of each message. Specifically, the matrix $\mathcal{M}_{T,:,:} \in \mathbb{R}^{|\mathcal{V}|\times d}$, a slice of $\mathcal{M}$, represents all messages in track T, and the vector $\mathcal{M}_{T,v,:} \in \mathbb{R}^d$, a fiber of $\mathcal{M}$, represents the message at node v in track T.

The initial messages $\mathcal{M}^{(0)}$ are constructed by loading node features $\mathbf{X}$ onto corresponding tracks in the loading step, which is given by, for each node $v \in \mathcal{V}$ and track $T \in \mathcal{T}$,

$$\begin{aligned} \mathcal{M}^{(0)}_{T,v,:} &= g(\mathbf{X}_{v,:}) \quad \text{if} \quad \mathbf{F}_{T,v} = 1 \\ \mathcal{M}^{(0)}_{T,v,:} &= \overrightarrow{\mathbf{0}} \qquad \text{if} \quad \mathbf{F}_{T,v} = 0, \end{aligned} \tag{1}$$

where the (0,1)-matrix $\mathbf{F} \in \{0,1\}^{|\mathcal{T}|\times|\mathcal{V}|}$ characterizes the node-track affiliations. Specifically, each $\mathbf{F}_{T,v} = 1$ indicates node v is affiliated with track T, by which nodes belonging to the same category are expected to be affiliated with the same track. An affine transformation $g : \mathbb{R}^m \to \mathbb{R}^d$ maps the m-dimensional node feature to d-dimensional message space. The d-dimensional zero vector $\overrightarrow{\mathbf{0}}$ indicates a blank message in tracks with which the node is not affiliated.

Following the loading, the multi-track message passing updates messages by L layers, *i.e.,* $\mathcal{M}^{(0)} \to \mathcal{M}^{(L)}$. Specifically, in each ℓ-th layer, messages in different tracks $T \in \mathcal{T}$ are independently propagated and aggregated. And for each track T, the message passing is defined as:

$$\mathcal{M}^{(\ell)}_{T,:,:} = (\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2})\mathcal{M}^{(\ell-1)}_{T,:,:} + \alpha_\ell \mathcal{M}^{(0)}_{T,:,:}, \tag{2}$$

where messages $\mathcal{M}^{(\ell-1)}_{T,:,:}$ in neighborhood are aggregated according to the normalized adjacency matrix $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. Additionally, we adopt two strategies in the message passing layer to facilitate a deeper graph model: (i) incorporating initial residual connections to the initial messages (Chen et al., 2020b), where α_l denotes a trade-off hyperparameter to control the integration of the initial messages; and (ii) omitting learnable weights $\mathbf{W}^{(\ell)}$ and activation function $\sigma(\cdot)$ in each layer of vanilla GCNs, to reduce the negative impact of excessive parameters on semi-supervised graph learning, inspired by the SGC (Wu et al., 2019).

Once completing the L-layer multi-track message passing, nodes acquire the updated messages $\mathcal{M}^{(L)}$ in their affiliated tracks to construct node representation $\mathbf{Z}$. For each node $v \in \mathcal{V}$, its representation vector $\mathbf{Z}_{v,:}$ is constructed by

$$\mathbf{Z}_{v,:} = (\sum\nolimits_{T\in\mathcal{T}} \mathbf{F}_{T,v} \cdot \mathcal{M}^{(L)}_{T,v,:})\mathbf{W}_Z, \tag{3}$$

where $\mathbf{W}_Z$ denotes a learnable linear transformation matrix. Notably, each node v only acquires messages in its affiliated tracks, *i.e.,* messages in track T that $\mathbf{F}_{T,v} = 1$. These node representations $\mathbf{Z}$ are subsequently used in downstream tasks, such as node classification in this paper.

3.2. Why MTMP Gains Improvements

3.2.1. Graph Learning Perspective

Preventing heterophily mixing. Firstly, MTMP gains improvements by preventing heterophilic mixing, which is crucial for addressing both the oversmoothing and over-squashing issues. Specifically, the MTMP layers in Eq. 2 only allow interactions between nodes of the same category when a perfectly correct node-track affiliation matrix $\mathbf{F}$ is available. In this way, the semantic purity and distinctiveness of node representations are guaranteed. Thus, the following proposition naturally holds.

Proposition 3.2. *If only nodes belonging to the same category are affiliated with a common track by* $\mathbf{F}$*, the multi-track message passing can prevent heterophily mixing.*

We empirically demonstrate the ability of MTMP to prevent heterophilic mixing by creating a perfect $\mathbf{F}$ in an idealized scenario where ground-truth labels are available. In this idealized scenario, node representations $\mathbf{Z}$ obtained by MTMP are visualized in a 2D space by t-SNE (Maaten & Hinton, 2008), as shown in Fig. 2A. In the figure, the majority of nodes with the same label exhibit spatial clustering. Only a few nodes appear outside their clusters, which is attributed to their exclusion from the maximal connected subgraph in Cora, thereby limiting their interaction with other nodes.

We further extend the above analysis to non-idealized scenarios, where $\mathbf{F}$ is imperfect, *i.e.,* part of nodes belonging to different categories are affiliated with the same track. We evaluate the performance of MTMP in conventional semi-supervised node classification on Cora, given $\mathbf{F}$ matrices

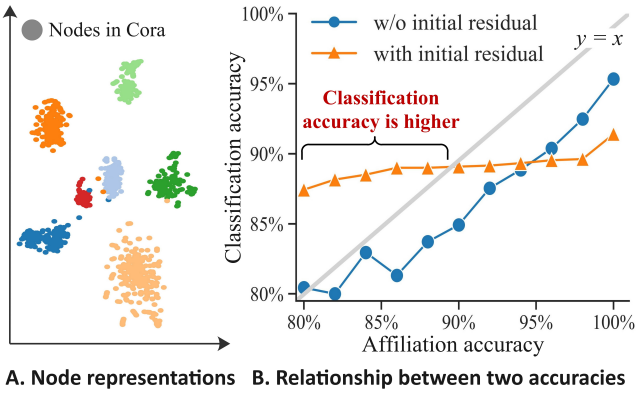

Figure 2. (A). Node representations obtained by MTMP using perfectly correct node-track affiliations $\mathbf{F}$. (B). The relation between classification accuracy of MTMP and affiliation accuracy of $\mathbf{F}$.

with varying affiliation accuracy. The affiliation accuracy is defined as the proportion of nodes correctly affiliated. The relationship between classification accuracy and affiliation accuracy is shown in Fig. 2B. We have two observations: (i) There exists a positive correlation between the two accuracies; (ii) MTMP can achieve higher accuracy than the affiliation accuracy by using initial residual connections. *These observations lead to the conclusion that a powerful MTGCN should combine MTMP with initial residual connections and equip with accurate affiliations* $\mathbf{F}$.

Facilitating long-distant information flow. Secondly, MTMP gains improvements by facilitating long-distant information flow to learn long-range dependences, which benefits in tackling oversquashing (Di Giovanni et al., 2023). We show this point by the following comparative analysis.

In existing MPNNs, message passing is inherently tied to the fusion of messages into a node's feature. Specifically, *message cannot pass through a node unless they are aggregated and fused into the node's feature*. If a message rejects to fuse into the node's feature, such as in gated GCNs (Li et al., 2015), the message is blocked by the node. This renders long-distant information flow very difficult.

In contrast, our proposed MTMP decouples messages from node representations in the message passing process. Messages exist in multiple tracks, enabling them to pass through any node regardless of whether they contribute to that node's feature. *This decoupling eliminates information loss due to heterophilic mixing in long-distant information flow.*

3.2.2. Semi-supervised Learning Perspective

Enhancing separation condition. Thirdly, MTMP gains improvements by enhancing separation condition, which is significant for effective semi-supervised learning according to theoretical study (Wei et al., 2020). We transfer the separation conditions in (Wei et al., 2020) to the context of graph topology to analyze the improvement of MTMP.

Definition 3.3. Given a graph $\mathcal{G}$, a message passing schema is $\mathcal{N}_k$*-separated* with probability $1-\mu$, if $R_{\mathcal{N}_k} \leq \mu$. Here, $R_{\mathcal{N}_k}$ denotes the proportion of nodes, each of which exchanges message with at least one k-hop neighbor belonging to a different category from its own category.

Here $\mathcal{N}_k$ denotes the k-hop neighborhood of a node in $\mathcal{G}$. According to Theorem 3.6 and 4.2 in (Wei et al., 2020), separation condition μ, roughly speaking, has a significant impact on the model's error bound. The lower μ, the lower error bound. Taking 1-hop neighborhood $\mathcal{N}_1$ as a case, in vanilla MPNNs, since each node exchanges messages with its all 1-hop neighbors, μ is the proportion of boundary nodes, *i.e.,* nodes whose labels differ from those of their neighbors. While in our proposed MTMP, nodes only exchange messages within each track, not across tracks. In this case, μ refers to the proportion of nodes incorrectly affiliated with tracks, *i.e.,* the error rate of node-track affiliations. To enable a quantitative comparison, we leverage classification results of a simple 2-layer GCN (Kipf & Welling, 2017) as node-track affiliations in MTMP. We compare the μ in vanilla MPPNS and MTMP on three citation graphs, as shown in Table 1. One can see separation conditions μ in MTMP are both significantly lower compared to those in vanilla MPNNs, suggesting that MTMP has lower error bounds in theory on the three graphs.

Table 1. Statistics of separation conditions μ

Datasets	Cora	Citeseer	Pubmed
μ in vanilla MPNNs	0.34	0.40	0.35
μ in our proposed MTMP	0.19	0.29	0.20

3.3. Node-Track Affiliations

The node-track affiliation matrix $\mathbf{F}$ plays a crucial role in our model, as previously analyzed. We leverage dot product attention (Vaswani et al., 2017) to obtain the $\mathbf{F}$. Specifically, the affiliation of node v is given by

$$\mathbf{F}_{:,v} = \mathrm{softmax}(\mathbf{H}_{v,:}\mathbf{W}_K(\mathbf{P}\mathbf{W}_Q)^{\mathrm{T}}). \tag{4}$$

Here, the vector $\mathbf{F}_{:,v} \in \mathbb{R}^{|\mathcal{T}|}$ represents the affiliation degree of node v to all tracks, and the vector $\mathbf{H}_{v,:}$ denotes the auxiliary representation of node v, $|\mathcal{T}|$-row matrix $\mathbf{P}$ represents the prototypes of all tracks, and $\mathbf{W}_K$ and $\mathbf{W}_Q$ are learnable parameters. The $\mathrm{softmax}$ operation is employed to generate a soft $\mathbf{F}$, which significantly eases parameter optimization by gradient. Experimental results on real-world graphs show the soft $\mathbf{F}$ closely approximates a rigid (0,1)-matrix. The process for calculating $\mathbf{F}$ is shown in Fig 3B.

The auxiliary representation of nodes $\mathbf{H}$ is generated by an auxiliary model Ψ. The primary purpose for $\mathbf{H}$, in calculating $\mathbf{F}$, is to contain distinct information of nodes from their ego-graphs. Notably, we need not pursue a model Ψ with

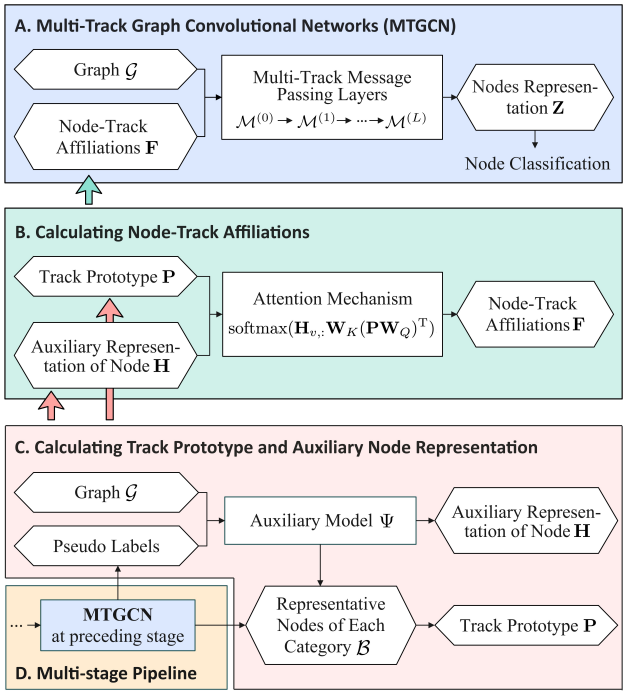

Figure 3. The framework of our proposed MTGCN.

extremely high performance. Even a simple GCN model can theoretically achieve improvements, as illustrated in Table 1. To show the efficacy of MTMP, we employ the simple 2-layer GCN (Kipf & Welling, 2017) as our auxiliary model Ψ. The model Ψ is trained using both training set and pseudo labels generated by MTGCN at the preceding stage, as shown in Fig. 3C and D.

The track prototypes $\mathbf{P}$ are constructed using representative nodes. Specifically, the prototype of track T is defined as

$$\mathbf{P}_{T,:} = \frac{1}{\Delta} \sum\nolimits_{v \in \mathcal{B}} \delta(y_v, T) \cdot \mathbf{H}_{v,:}. \tag{5}$$

Here, the set $\mathcal{B}$ comprises representative nodes v, including labeled nodes in training set and unlabeled nodes that are likely to be correctly predicted by the auxiliary model Ψ and MTGCN model in the preceding stage (see subsection 3.4). The likelihood is measured by *softmax confidence score*, and powerful measures can be easily integrated (Pei et al., 2024b). Besides, the delta function $\delta(y_v, T) \in \{0, 1\}$ indicates whether the label (for labeled nodes) or pseudo-label (for unlabeled nodes) of node v aligns with the category corresponding to track T. $\Delta = \sum_{v \in \mathcal{B}} \delta(y_v, T)$ is a normalization factor. Aggregating representative nodes of each category as a track prototype ensures that category semantics are riched in the track prototypes. The process of MTGCN is presented as an algorithm in the Appendix.

3.4. Multi-stage Training Pipeline

We design a multi-stage training pipeline for MTGCN. In each stage, MTGCN is retrained using training set along with useful information obtained from the MTGCN at the preceding stage. The information includes pseudo labels and representative nodes, as shown in Fig. 3D. This multi-stage pipeline is a self-evolutionary strategy, enriching the MTGCN with progressively acquired information and providing opportunities to escape local optima. The multi-stage training algorithm is presented in the Appendix.

We outline the training process for each stage. We first train the auxiliary model Ψ using training set and pseudo labels. We then calculate auxiliary representations $\mathbf{H}$ and track prototypes $\mathbf{P}$ by using auxiliary model Ψ and the MTGCN at the preceding stage. With $\mathbf{H}$ and $\mathbf{P}$, we optimize parameters in attention module and MTMP by using a cross-entropy loss based on only labeled nodes in data. Additionally, we design a constraint to facilitate the optimization of parameters in the attention module, $\mathcal{L}_R = \Sigma_{v \in \mathcal{B}} ||\mathbf{F}_{:,v} - \text{one-hot}(y_v)||_F$. The constraint enforces that the representative nodes in $\mathcal{B}$ can be affiliated with correct tracks based on their lables or pseudo labels y_v.

4. Experiments

In this section, we first validate the proposed MTGNN on several real-world graphs, and then demonstrate the ability of MTGCN to tackle oversmoothing and oversquashing issues. Finally, we empirically analyze the node-track affiliations. The hyper-parameter settings in MTGCN and dataset descriptions are presented in the Appendix.

4.1. Node Classification on Real-world Graphs

4.1.1. Semi-supervised Node Classification

The MTGCN is first validated in semi-supervised node classification task on homophilic graphs, including Cora (McCallum et al., 2000), Citeseer (Sen et al., 2008), Pubmed (Namata et al., 2012), and Coauthor (CS and Physics) (Yang et al., 2016). For all graphs, we use 20 nodes per class for training, 500 validation nodes, and 1,000 testing nodes. For the three citation graphs, we use the standard training/validation/testing split provided in (Yang et al., 2016; Kipf & Welling, 2017). We compare MTGCN with recent baseline models including GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018),Self-train (Li et al., 2018), DisenGCN (Ma et al., 2019), GCNII (Chen et al., 2020b), EGNN (Zhou et al., 2021b), PDE-GCN (Eliasof et al., 2021), GRAND++(Thorpe et al., 2022), GraphCON (Rusch et al.), ACMP (Wang et al., 2023), GREAD (Choi et al., 2023). In all experiments, we run MTGCN 10 times with different initializations and report the average classification accuracies in Table 2. The "-sn" suffix denotes the MTGCN model at stage n, and the dash symbol "-" denotes that the result is not available in the original paper.

The experimental results in Table 2 demonstrate the effectiveness of our proposed MTGCN in semi-supervised

Table 2. Comparisons of node classification accuracy in semi-supervised setting (%). The best two models are emphasized in red (best) and blue (second best).

Datasets	Cora	Cite.	Pubm.	Co.CS	Co.Phys
GCN	80.01	70.41	79.01	90.01	93.81
GAT	81.21	70.81	78.52	91.13	93.31
Self-Train	82.27	73.24	80.32	-	-
DisenGCN	83.30	72.44	80.30	90.96	94.28
GCNII	85.30	73.10	80.10	88.50	93.90
EGNN	85.70	-	80.10	-	93.30
GRAND++	83.60	73.40	78.80	-	-
PDE-GCN	84.30	75.60	80.60	-	-
GraphCON	84.20	74.20	79.40	-	-
ACMP	84.91	73.75	79.01	84.02	93.47
GREAD	84.72	73.31	78.17	88.52	92.24
MTGCN-s1	85.00	73.33	80.31	87.61	94.30
MTGCN-s2	85.97	73.35	81.10	92.15	94.57
MTGCN-s3	86.40	74.60	80.92	91.57	94.55
MTGCN-s4	85.44	73.88	80.33	92.54	94.72

setting. Specifically, we have the following two observations: (i) In most cases, the classification accuracies of MTGCN are higher than baseline models on the five graph datasets, which indicates that MTGCN is capable of learning effective node representations for node classification. (ii) The MTGCN-s3 and MTGCN-s4 usually exhibit the best performance, which demonstrates the effectiveness of the multi-stage training pipeline for MTGCN, especially on the CoauthorCS dataset. In other words, the MTGCN is self-evolutionary during multiple stages.

4.1.2. Fully-supervised Node Classification

We further validate the MTGCN in fully-supervised node classification task. The experiments are conducted on three homophilic graphs, including Cora, Citeseer, Pubmed, and three heterophilic graphs, including Cornell, Texas, and Wisconsin (Pei et al., 2020). For all graphs, we set the train/validation/test splits as 48%, 32%, and 20%, respectively. In addition to comparing with GCN, GAT, and GCNII, we compare MTGCN with models specially designed for heterophilic graphs, including H2GCN (Zhu et al., 2020), GemoGCN (Pei et al., 2020), LINKX (Lim et al., 2021), GGCN (Yan et al., 2022), GRAFF (Di Giovanni et al., 2022), Sheaf (Bodnar et al., 2022), ACM-GCN (Luan et al., 2022), Half-hop (Azabou et al., 2023).

Experimental results are summarized in Table 3. The results demonstrate the effectiveness of our proposed MTGCN in fully-supervised setting. Specifically, we have the following two observations: (i) The MTGCN achieves outstanding performance on the three heterophilic graphs, and it outperforms all the baselines. The performance can be largely attributed to MTGCN's ability to prevent heterophily mixing during message passing, thereby capturing long-distant

Table 3. Comparisons of node classification accuracy in full-supervised setting (%). The best two models are emphasized in red (best) and blue (second best).

Datasets Homophily	Cora 0.81	Cite. 0.80	Pubm. 0.74	Corn. 0.30	Texas 0.11	Wisc. 0.21
GCN	85.77	73.68	88.13	52.70	52.16	48.92
GAT	86.37	74.32	87.62	54.32	58.38	49.41
GCNII	88.49	77.13	90.30	74.86	69.46	74.12
GeomGCN	85.27	77.99	90.05	60.81	67.57	64.12
LINKX	84.64	73.19	87.86	77.84	74.60	75.49
GGCN	87.95	77.14	89.15	85.68	84.86	86.86
H2GCN	87.87	77.11	89.49	82.70	84.86	87.65
ACM-GCN	88.25	77.12	89.71	85.95	86.76	87.45
Sheaf	86.90	76.70	89.49	84.86	85.05	89.41
GRAFF	87.61	76.92	88.95	83.24	88.38	87.45
Half-hop	83.48	71.40	88.15	72.36	69.21	70.78
GraphCON	88.03	74.96	86.43	84.30	85.40	87.80
MTGCN-s1	90.61	76.46	88.43	84.21	84.21	90.20
MTGCN-s2	89.68	77.06	88.11	86.84	92.10	88.23
MTGCN-s3	90.42	77.36	88.26	86.84	89.47	90.20
MTGCN-s4	90.60	76.91	88.01	89.47	92.10	90.20

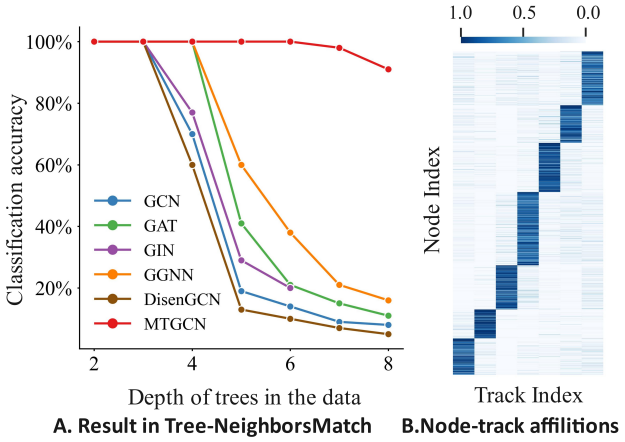

Figure 4. (A). MTGCN achieves nearly perfect accuracy on the Tree-NeighborsMatch problem. (B). Visualization of node-track affiliation matrix from MTGCN trained on the Cora dataset.

dependencies and addressing oversmoothing in heterophilic graphs. (ii) Compared to semi-supervised setting, the performance gap between MTGCNs at different stages becomes less, especially in homophilic graphs. This can be largely attributed to the sufficient supervision information available in the fully-supervised setting. That is, the multi-stage training is more critical in semi-supervised graph learning.

4.2. Analysis on Oversmoothing and Oversquashing

4.2.1. Tackling Oversmoothing

We validate the capacity of MTGCN to address oversmoothing. The experiments are conducted on the three citation graphs and follow the same experimental setup as that in semi-supervised node classification in Section. 4.1.1. We compare MTGCN with models specially designed to mit-

Table 4. Semi-supervised node classification accuracy (%) and group distance ratio R_g across various model depth.

Dataset	Cora						Citeseer						Pubmed					
# of layers	2	4	8	16	32	64	2	4	8	16	32	64	2	4	8	16	32	64
GCN	80.0	80.4	69.5	64.9	60.3	28.7	70.8	67.6	30.2	18.3	25.0	20.0	79.0	76.5	61.2	40.9	22.4	35.3
GAT	81.2	79.8	62.3	31.9	31.9	14.9	70.8	67.0	48.5	23.1	23.1	18.1	78.6	76.9	76.5	41.3	41.3	40.7
DropEdge	82.8	82.0	75.8	75.7	62.5	49.5	72.3	70.6	61.4	57.2	41.6	34.4	79.6	79.4	78.1	78.5	77.0	61.5
JKNet	-	80.2	80.7	80.2	81.1	71.5	-	68.7	67.7	69.8	68.2	63.4	-	78.0	78.1	72.6	72.4	74.5
Incep	-	77.6	76.5	81.7	81.7	80.0	-	69.3	68.4	70.2	68.0	67.5	-	77.7	77.9	74.9	-	-
GCNII	80.2	82.3	82.8	83.5	84.9	85.3	66.1	66.7	70.6	72.0	73.2	73.1	77.7	78.2	78.8	80.3	79.8	80.1
PDE-GCN	82.0	83.6	84.0	84.2	84.3	84.3	74.6	75.0	75.2	75.5	75.6	75.5	79.3	80.6	80.1	80.4	80.2	80.3
DisenGCN	77.6	83.3	82.7	82.9	82.2	69.1	70.1	69.3	71.3	72.2	70.6	65.4	76.4	76.5	80.3	78.8	76.6	75.0
MTGCN	80.5	83.4	84.9	86.2	85.9	86.4	70.1	72.8	72.9	74.6	73.8	74.0	78.7	80.7	80.5	80.8	81.0	81.1
R_g of MTGCN	0.249	0.313	0.368	0.383	0.382	0.381	0.293	0.328	0.368	0.383	0.383	0.382	0.837	0.918	1.031	1.076	1.035	1.027

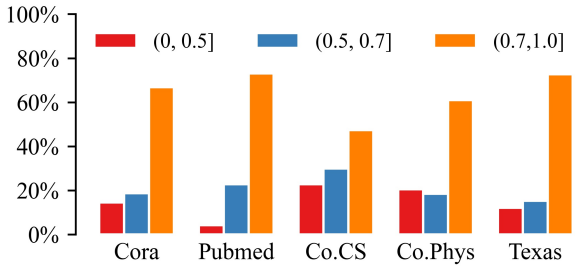

Figure 5. The distribution of nodes in terms of their affiliation strength. More than half of the nodes show a strong affiliation to a single track, with an affiliation strength exceeding 0.7.

igate oversmoothing, including DropEdge (Rong et al., 2019), JKNet (Xu et al., 2018), Incep (Kazi et al., 2019), GCNII, and PDE-GCN. We adopt two oversmoothing measures, classification accuracy and group distance ratio R_g (Zhou et al., 2020). The ratio R_g characterizes the ratio between inter-group distance and intra-group distance of node representations. The lower R_g, the higher oversmoothing. A detailed definition of R_g is provided in the Appendix. We calculate classification accuracies and R_g of models with varying layers, ranging from 2 to 64.

Experimental results, summarized in Table 4, illustrate the effectiveness of our proposed MTGCN in addressing oversmoothing. Specifically, we have the following two observations: (i) MTGCN maintains stable classification accuracies and R_gs as the number of layers increases, signifying its capacity to oversmoothing. In contrast, GCN, GAT, DropEdge, JKNet, and DisenGCN exhibit noticeable declines in performance beyond 32 layers, indicating susceptibility to oversmoothing. (ii) We observe a general trend of gradual improvement in classification accuracies as the number of layers in MTGCN increases. This trend suggests that deeper MTGCN can learn more information from the graphs, which is enlightening for developing deep graph models in future.

4.2.2. Tackling Oversquashing

We also validate MTGCN's effectiveness in addressing oversquashing through experiments on a synthetic benchmark specifically crafted for this purpose: the Tree-NeighborsMatch problem (Alon & Yahav, 2021). The benchmark is an inductive node classification task on 208,192 trees with different depths. The higher the training accuracy on this task, the more effectively models overcoming the oversquashing issue.

The experimental results in Fig. 4A show MTGCN can effectively address over-squashing, achieving nearly perfect training accuracy. We have two additional observations: (i) Beyond tree depth of 5, the training accuracy of all models, except for MTGCN, markedly declines, indicative of their struggle with oversquashing. (ii) MTGCN experiences a slight decrease in accuracy when tree depth exceeds 6 layers. This decrease is attributed to the high space complexity of MTGCN, which is further discussed in Section 6.

Table 5. Accuracy of node-track affiliations $\mathbf{F}$ (%)

Dataset	Cora	Pubmed	Co.CS	Co.Phys	Texas
Accuracy	73.85	76.37	77.05	85.06	65.03

4.3. Analysis on Node-Track Affiliations

We empirically analyze the node-track affiliation matrix $\mathbf{F}$ estimated from graph datasets. We first visualize $\mathbf{F}$ from MTGCN trained on Cora, sorting nodes according to their most likely affiliated track. As illustrated in Fig. 4 B, the $\mathbf{F}$ matrix exhibits a block diagonal pattern, which indicates that each node is highly likely to affiliate with a single track and is unlikely to be associated with other tracks. This suggests that *MTGCN is not an ensemble of different tracks but facilitates independent message propagation on separate tracks, effectively preventing heterophilic mixing.* To further quantify this, we calculate distribution of nodes in terms of affiliation strength, $\mathbf{F}_{T,v}$ value on five graphs, as shown in Fig. 5. In the figure, more than half of nodes have a strong affiliation to a track, with an affiliation strength exceeding 0.7, which further supports the above conclusion.

Additionally, we report affiliation accuracies on the five graphs in Table 5. A considerable proportion of nodes are incorrectly affiliated, probably because we just adopt a simple 2-layer GCN as the auxiliary model Ψ, limiting the quality of auxiliary node representations. It is worth noting

that despite the imperfect node-track affiliations, MTGCN has shown significant improvements on benchmark graphs. This suggests the performance of MTGCN might be further improved if it is equipped with a more powerful Ψ.

5. Related Work

5.1. Tackling Oversmoothing and Oversquashing

Graph rewiring. Graph rewiring optimizes graph topology to migrate oversmoothing and oversquashing. However, it potentially damages the patterns of original graph topology.

Removing inter-class edges or even randomly removing edges during training is an intuitive approach to alleviate oversmoothing (Chen et al., 2020a; Rong et al., 2019; Hasanzadeh et al., 2020). Ollivier-Ricci curvature can guide edge removal for alleviating oversmoothing and oversquashing (Nguyen et al., 2023). Supervised information is also used to learn the removal of task-irrelevant edges that cause oversmoothing in heterophilic graphs (Zheng et al., 2020; Yan et al., 2022). Besides edge removal, PairNorm (Zhao & Akoglu, 2019) sets pairwise distances to be constant, and Half-Hop (Azabou et al., 2023) introduces "slow nodes" to enhance message passing, both targeting oversmoothing.

Adding edges between distant nodes for reducing commute time is typically used to alleviate oversquashing (Brüel-Gabrielsson et al., 2022; Abboud et al., 2022; Bodnar et al., 2021). Due to the connection between commute times and curvature (Devriendt & Lambiotte, 2022), the edges to mitigate bottlenecks of MPNNs can be identified by negative curvature (Topping et al., 2022). Effective resistance between nodes has been used to guide the edge addition (Black et al., 2023). Furthermore, the spectral gap has been identified as a critical factor in oversquashing (Banerjee et al., 2022), which leads to approaches that identify the adding edges by optimizing the spectral gap (Karhadkar et al., 2022; Arnaiz-Rodríguez et al., 2022). Recent research in (Arnaiz-Rodríguez et al., 2022) reveals the relationship between commute time and spectral gap. Graph transformers, which integrate a complete graph with weights via attention, are a special case of adding edges (Kreuzer et al., 2021; Rampášek et al., 2022; Brüel-Gabrielsson et al., 2022).

Regularization. Constraining node representations to be distinctive during training can effectively prevent oversmoothing, by employing Dirichlet energy (Zhou et al., 2021b), group normalization (Zhou et al., 2020), NodeNorm (Zhou et al., 2021a), and *etc*. Regulating information flow in message passing is another strategy. This can be achieved by incorporating gating (Bresson & Laurent, 2017) or gradient gating (Rusch et al., 2022) into GCNs. The information flows in multi-channels are regulated as orthogonal to each other to prevent oversmoothing (Yang et al., 2022; 2023). Additionally, several physics-inspired models, such as oscillator networks (Rusch et al.), Allen-Cahn message passing (Wang et al., 2023), and gradient flow (Di Giovanni et al., 2022), are inherent constraints. However, these regularizations may degrade model performance.

Residual connection. Residual connections are used to mitigate oversmoothing (Xu et al., 2018; Liu et al., 2020a), particularly initial residual connections (Chen et al., 2020b; Gasteiger et al., 2019). Recently, geometric skip connection is proposed to tackle oversquashing (Gutteridge et al., 2023).

5.2. Multi-channel Graph Convolutional Network

To learn disentangled representations of graphs, the disentangled multi-channel convolutional layer and neighborhood routing mechanism are proposed in DisenGCN (Ma et al., 2019). Tacking DisenGCN as a base model, kernel Hilbert-Schmidt independence criterion (Liu et al., 2020b), contrastive learning (Li et al., 2021), and diversity regularizer (Guo et al., 2022) are proposed and integrated to further enhance disentanglement. In these multi-channel models, the neighborhood routing functions as a specialized version of MTMP that operates within local neighborhoods. This allows for locally preventing of heterophilic mixing. Unlike MTMP, these models do not decouple messages from node representations, which limits their ability to prevent heterophilic mixing on a non-local scale. As shown in our experiments, DisenGCN fails to address the oversquashing.

In summary, the MTGCN is fundamentally distinct from these existing approaches, as it incorporates MTMP, a novel message passing schema with ability to prevent heterophilic mixing. This unique feature enables MTGCN to tackle both oversmoothing and oversquashing issues in graph learning.

6. Model Limitation — No Free Lunch

Compared with vanilla GCNs, our proposed MTGCN requires extra storage space to process messages $\mathcal{M}$ in multiple tracks, resulting in a space complexity of $\mathcal{O}(d|\mathcal{T}||\mathcal{V}|\eta)$. Here, d denotes the dimension of each message, $|\mathcal{T}|$ the number of tracks, $|\mathcal{V}|$ the number of nodes in graph, and η the number of graphs. For large graphs with numerous nodes, we can circumvent the issue of high space complexity by adopting training on smaller subgraphs. However, addressing high space complexity becomes challenging when the number of tracks, $|\mathcal{T}|$, increases significantly. In such cases, storing messages $\mathcal{M}$ could lead to memory overflow errors, thereby constraining the applicability of MTGCN. This limitation is observed in the experiments shown in Fig. 4 A, in which the performance of MTGCN exhibits a slight decline beyond a depth of seven layers. Considering a depth of seven layers, the space complexity $\mathcal{O}(d|\mathcal{T}||\mathcal{V}|\eta)$ becomes 1.67×10^{10}, as there are 32,000 graphs and 128 categories, posing challenges for model training.

7. Conclusion

In this paper, we investigate into deeper graph neural networks and argue that the prevalent challenges of oversmoothing and oversquashing in graph learning stem from heterophilic mixing in aggregation. To overcome these challenges, we introduced a novel multi-track graph convolutional network (MTGCN) specifically designed to counteract heterophilic mixing. The core of MTGCN is a multi-track message passing (MTMP) scheme, which propagates and aggregates messages in respective tracks so that maintains the semantic purity of messages and outputs distinctive node representation vectors. Through empirical validation, MTGCN demonstrated outstanding performance, successfully addressing oversmoothing and oversquashing. As future work, we will explore improve affiliation accuracy by choosing a right auxiliary model — depending not only on input graphs but also on target applications, such as infection control on social contact network (Pei et al., 2022).

Impact Statement

This paper introduces a new GNN model - the Multi-Track Message Graph Convolutional Network (MTGCN), aiming to address the issues of oversmoothing and oversquashing in graph learning. By establishing message tracks, MTGCN independently propagates and aggregates messages according to the category semantics. This approach effectively avoids heterogeneous mixing, maintains the semantic purity of messages, and learns better node representations. The main impact of this paper is MTGCN offers a novel direction — multi-track message passing — for developing deeper GNNs.

Additionally, MTGCN is expected to have a positive impact on applications that utilize graph-structured data, such as recommendation systems, computational chemistry, and social networks. Its enhanced capability in handling graphs will enable these applications to achieve more accurate and efficient graph analysis. As MTGCN finds applications in various fields, it is crucial to consider its ethical and social impacts. Ensuring that the development and use of this technology comply with ethical standards and are socially responsible is of paramount importance.

Acknowledgements

The authors would like to thank all the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China under grant 62202369, U22B2019, 62372362, and 62306229.

References

Abboud, R., Dimitrov, R., and Ceylan, I. I. Shortest path networks for graph property prediction. In *The First Learning on Graphs Conference*, 2022.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Arnaiz-Rodríguez, A., Begga, A., Escolano, F., and Oliver, N. M. Diffwire: Inductive graph rewiring via the lovász bound. In *The First Learning on Graphs Conference*, 2022.

Azabou, M., Ganesh, V., Thakoor, S., Lin, C.-H., Sathidevi, L., Liu, R., Valko, M., Veličković, P., and Dyer, E. L. Half-hop: A graph upsampling approach for slowing down message passing. In *International Conference on Machine Learning*, 2023.

Banerjee, P. K., Karhadkar, K., Wang, Y. G., Alon, U., and Montúfar, G. Oversquashing in gnns through the lens of information contraction and graph expansion. In *Annual Allerton Conference on Communication, Control, and Computing*, 2022.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Black, M., Wan, Z., Nayyeri, A., and Wang, Y. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning*, 2023.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lio, P., and Bronstein, M. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, 2021.

Bodnar, C., Di Giovanni, F., Chamberlain, B., Liò, P., and Bronstein, M. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 2022.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Brüel-Gabrielsson, R., Yurochkin, M., and Solomon, J. Rewiring with positional encodings for graph neural networks. *arXiv preprint arXiv:2201.12674*, 2022.

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

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020b.

Choi, J., Hong, S., Park, N., and Cho, S.-B. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*, 2023.

Devriendt, K. and Lambiotte, R. Discrete curvature on graphs from the effective resistance. *Journal of Physics: Complexity*, 3(2):025008, 2022.

Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*, 2022.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., and Bronstein, M. M. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, 2023.

Eliasof, M., Haber, E., and Treister, E. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in neural information processing systems*, 2021.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, 2017.

Giovanni, F. D., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. Graph neural networks as gradient flows: understanding graph convolutions via energy. In *International Conference on Learning Representations*, 2023.

Goller, C. and Kuchler, A. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks*, 1996.

Guo, J., Huang, K., Yi, X., and Zhang, R. Learning disentangled graph convolutional networks locally and globally. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Gutteridge, B., Dong, X., Bronstein, M. M., and Di Giovanni, F. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, 2023.

Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., and Qian, X. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*, 2020.

Karhadkar, K., Banerjee, P. K., and Montúfar, G. Fosr: First-order spectral rewiring for addressing oversquashing in gnns. *arXiv preprint arXiv:2210.11790*, 2022.

Kazi, A., Shekarforoush, S., Arvind Krishna, S., Burwinkel, H., Vivar, G., Kortüm, K., Ahmadi, S.-A., Albarqouni, S., and Navab, N. Inceptiongcn: receptive field aware graph convolutional network for disease prediction. In *Information Processing in Medical Imaging*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 2021.

Li, H., Wang, X., Zhang, Z., Yuan, Z., Li, H., and Zhu, W. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 2021.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 2021.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020a.

Liu, Y., Wang, X., Wu, S., and Xiao, Z. Independence promoted graph disentangled networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020b.

Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.-W., and Precup, D. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 2022.

Ma, J., Cui, P., Kuang, K., Wang, X., and Zhu, W. Disentangled graph convolutional networks. In *International conference on machine learning*, pp. 4212–4221, 2019.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Namata, G., London, B., Getoor, L., and Huang, B. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, 2012.

Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S., and Nguyen, T. M. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning*, 2023.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Pei, H., Yang, B., Liu, J., and Chang, K. C.-C. Active surveillance via group sparse bayesian learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1133–1148, 2022.

Pei, H., Chen, T., Chen, A., Deng, H., Tao, J., Wang, P., and Guan, X. Hago-net: Hierarchical geometric massage passing for molecular representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14572–14580, 2024a.

Pei, H., Xiong, Y., Wang, P., Tao, J., Liu, J., Deng, H., Ma, J., and Guan, X. Memory disagreement: A pseudo-labeling measure from training dynamics for semi-supervised graph learning. In *Proceedings of the ACM on Web Conference 2024*, pp. 434–445, 2024b.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 2022.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. Graph-coupled oscillator networks. In *International Conference on Machine Learning*.

Rusch, T. K., Chamberlain, B. P., Mahoney, M. W., Bronstein, M. M., and Mishra, S. Gradient gating for deep multi-rate learning on graphs. *arXiv preprint arXiv:2210.00513*, 2022.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Thorpe, M., Nguyen, T. M., Xia, H., Strohmer, T., Bertozzi, A., Osher, S., and Wang, B. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representation*, 2022.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, H., Lian, D., Tong, H., Liu, Q., Huang, Z., and Chen, E. Hypersorec: Exploiting hyperbolic user and item representations with multiple aspects for social-aware recommendation. *ACM Transactions on Information Systems (TOIS)*, 40(2):1–28, 2021.

Wang, Y., Yi, K., Liu, X., Wang, Y. G., and Jin, S. ACMP: Allen-cahn message passing with attractive and repulsive forces for graph neural networks. In *International Conference on Learning Representations*, 2023.

Wei, C., Shen, K., Chen, Y., and Ma, T. Theoretical analysis of self-training with deep networks on unlabeled data. In *International Conference on Learning Representations*, 2020.

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 2018.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., and Koutra, D. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, 2022.

Yang, L., Kang, L., Zhang, Q., Li, M., He, D., Wang, Z., Wang, C., Cao, X., Guo, Y., et al. Open: Orthogonal propagation with ego-network modeling. *Advances in Neural Information Processing Systems*, 35:9249–9261, 2022.

Yang, L., Zhang, Q., Shi, R., Zhou, W., Niu, B., Wang, C., Cao, X., He, D., Wang, Z., and Guo, Y. Graph neural networks without propagation. In *Proceedings of the ACM Web Conference 2023*, pp. 469–477, 2023.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Zhao, Q., Lindell, D. B., and Wetzstein, G. Learning to solve pde-constrained inverse problems with graph networks. *arXiv preprint arXiv:2206.00711*, 2022.

Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., Chen, H., and Wang, W. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, 2020.

Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., and Hu, X. Towards deeper graph neural networks with differentiable group normalization. *Advances in neural information processing systems*, 2020.

Zhou, K., Dong, Y., Wang, K., Lee, W. S., Hooi, B., Xu, H., and Feng, J. Understanding and resolving performance degradation in deep graph convolutional networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021a.

Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S.-H., and Hu, X. Dirichlet energy constrained learning for deep graph neural networks. *Advances in Neural Information Processing Systems*, 2021b.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 2020.

APPENDIX

This appendix consists of two sections. In section A, we provide a detailed explanation of the computation process and multi-stage training strategy of MTGCN. In section B, we describe experimental details.

A. Algorithms

We present the calculating process of MTGCN in Algorithm 1, in which the equation numbers correspond to the equations in the main paper. The multi-stage training algorithm is presented in the Appendix 2. This multi- stage pipeline is a self-evolutionary strategy, enriching the MTGCN with progressively acquired information and providing opportunities to escape local optima.

Algorithm 1 Calculating process in MTGCN

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: Node representations $\mathbf{Z}$

Initialize learnable parameters in MTGCN
Generate auxiliary node representations $\mathbf{H}$ by Ψ
Construct set $\mathcal{B}$ by identifying representative nodes
for each track $T \in \mathcal{T}$ **do**
  Calculate track prototype $\mathbf{P}_{T,:}$ using Eq. (5)
end for
Calculate node-track affiliations $\mathbf{F}$ using Eq. (4)
Obtain initial messages $\mathcal{M}^(0)$ using Eq. (1)
for $l = 1$ to L **do**
  Update messages $\mathcal{M}^l$ using Eq. (2)
end for
Calculate node representations $\mathbf{Z}$ by Eq. (3)

Algorithm 2 Multi-stage training for MTGCN

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: Node representations $\mathbf{Z}$
Parameter: Number of stage K

Initialize pseudo label set $\mathcal{Y}_{PL} \leftarrow \varnothing$ and representive node set $\mathcal{B} \leftarrow \varnothing$
for each stage $k = 1$ to K **do**
  Train auxiliary mode Ψ on $\mathcal{G}$ and $\mathcal{Y}_{PL}$
  Calculate auxiliary node representations $\mathbf{H}$ and Update set $\mathcal{B}$ by Ψ
  Update track prototype $\mathbf{P}$ by $\mathbf{H}$ and $\mathcal{B}$
  if $k < K$ **then**
    Calculate node representations $\mathbf{Z}$ by Algorithm 1 with the updated $\mathbf{H}$ and $\mathbf{P}$
    Update $\mathcal{Y}_{PL}$ and $\mathcal{B}$ according to updated $\mathbf{Z}$
  end if
end for
Calculate node representations $\mathbf{Z}$ by Algorithm 1 by Algorithm 1 with the updated $\mathbf{H}$ and $\mathbf{P}$

B. Experimental details

B.1. Dataset information

We consider two types of datasets: Homophilic and Heterophilic. They are differentiated by the homophily level of a graph (Pei et al., 2020)

$$\mathcal{H} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{(w, v) : w \in \mathcal{N}(v) \wedge y_v = y_w\}|}{|\mathcal{N}(v)|} \tag{6}$$

In the experiments, we use five homophilic datasets, including Cora (McCallum et al., 2000), Citeseer (Sen et al., 2008) and Pubmed (Namata et al., 2012), Coauthor (Yang et al., 2016), and three heterophilic datasets: Cornell, Texas, and Wisconsin from the WebKB dataset (Pei et al., 2020). We list the numbers of classes, features, nodes and edges of each dataset, and their homophily level in Table 6. A low homophily level suggests a more heterophilic dataset, where neighbors are often not from the same class. Conversely, a high homophily level indicates a dataset that is closer to homophilic, with similar nodes tending to be connected.

Table 6. Dataset statistics

Dataset	Classes	Features	# Nodes	# Edges	Homophily
Cora	7	1433	2708	5278	0.81
Citeseer	6	3703	3327	4552	0.74
Pubmed	3	500	19717	44324	0.80
Co.CS	15	6805	18333	81894	0.80
Co.Phys	5	8415	34493	247962	0.92
Texas	5	1703	183	309	0.11
Wisconsin	5	1703	251	499	0.21
Cornell	5	1703	183	499	0.30

B.2. Hyper-parameter settings

We use the Adam SGD optimizer (Kingma & Ba, 2014) with a learning rate of 0.01 and the early stopping strategy with a patience of 100 epochs to train MTGCN. All hyper-parameters for training MTGCN are listed in Table 7, including the total number of tracks ("# track"), the weight decay ("WD"), the dropout rate, the weight of the constraint $\mathcal{L}_R$ ("WR"), and the total number of pseudo-labeled nodes used in each category at each stage ("num_ks"). In all experiments, we set four training stages in Multi-stage training strategy. ALL hyper-parameters are obtained by grid search. The code of grid search can be found in our Github repository [2].

In addition, the configuration of the initial residual connections follows previous study (Chen et al., 2020b). All experiments are implemented in Python 3.8.13 with PyTorch Geometric on one NVIDIA Tesla V100 GPU.

[2]Code of grid search: https://github.com/XJTU-Graph-Intelligence-Lab/mtgcn

B.3. Definition of R_g

The group distance ratio R_g in Table 4 in the main paper is used to measure the degree of oversmoothing. Specifically, R_g is defined as the ratio between inter-group distance and intra-group distance of node representations.

$$R_g = \frac{C}{(C-1)^2} \frac{d_{inter}}{d_{intra}} \tag{7}$$

$$d_{inter} = \sum_{i \neq j} \frac{1}{|\boldsymbol{L}_i|\,|\boldsymbol{L}_j|} \sum_{h_{iv} \in \boldsymbol{L}_i} \sum_{h_{jv'} \in \boldsymbol{L}_j} \|h_{iv} - h_{jv'}\|_2) \tag{8}$$

$$d_{intra} = \sum_{i} \frac{1}{|\boldsymbol{L}_i|^2} \sum_{h_{iv}, h_{iv'} \in \boldsymbol{L}_i} \|h_{iv} - h_{iv'}\|_2 \,, \tag{9}$$

where $h_{i,v}$ denotes v's node representation vector, where node v is associated with the label i. Correspondingly, $\boldsymbol{L}_i$ denotes the group of representation vectors of all nodes in the i-th group (category), and C denotes the number of node groups (categories). In addition, $||\cdot||_2$ denotes the L2 norm of a vector and $|\cdot|$ denotes the set cardinality. In this way, a low R_g ratio means that node representations between different groups (categories) are indistinguishable, indicating a high oversmoothing.

Table 7. Hyper-parameter settings of MTGCN

Dataset	WR	WD	# track	dropout	num_ks
Cora	0.5	5e-4	7	0.5	100,50,20,1
CiteSeer	0.5	5e-4	6	0.5	100,50,20,1
PubMed	0.5	5e-4	3	0.5	400,50,20,1
Co.CS	0.5	1e-4	15	0.5	50,10,5,1
Co.Phys	0.5	1e-4	5	0.5	1500,1500,700,10
Texas	0.9	5e-4	5	0.3	5,2,2,1
Wisconsin	0.9	5e-4	5	0.3	5,2,2,1
Cornell	0.9	5e-4	5	0.3	5,2,2,1