Multi-Track Message Passing: Tackling Over-smoothing and Over-squashing in Graph Learning via Preventing Heterophily Mixing

Presenter: Yu Li

Xi'an Jiaotong University

July 2024

CONTENTS

Background: Message-Passing Neural Networks (MPNNs)

- MPNNs has achieved successes in various applications, from recommend-dation system to molecular dynamics analysis

Two Basic operations in MPNNs:

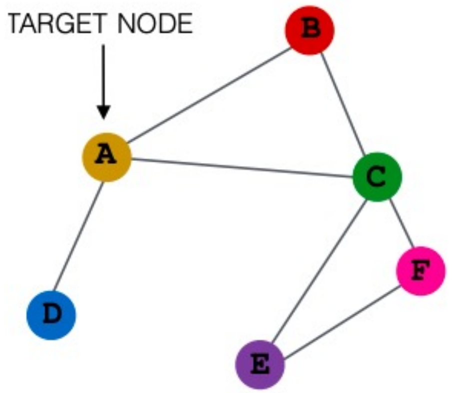
(1) Message generation : Message generation through self-representation of individual nodes

$$\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$$

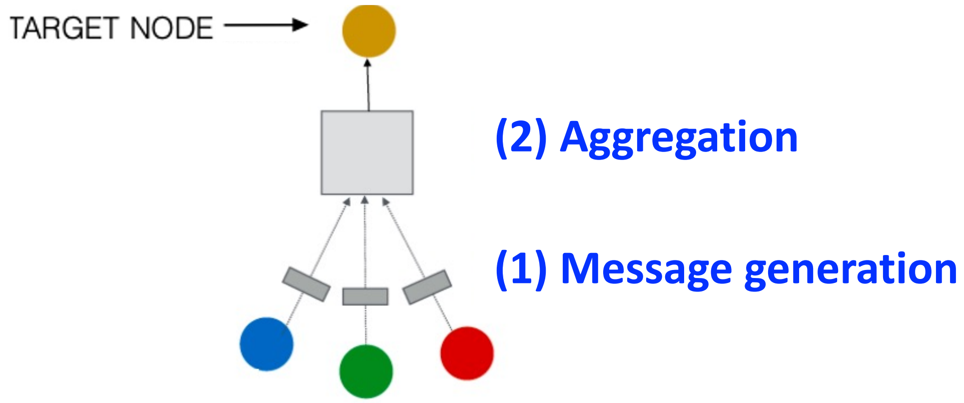
(2) Aggregation: Message aggregation from neighborhood

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)$$

Input: Graph



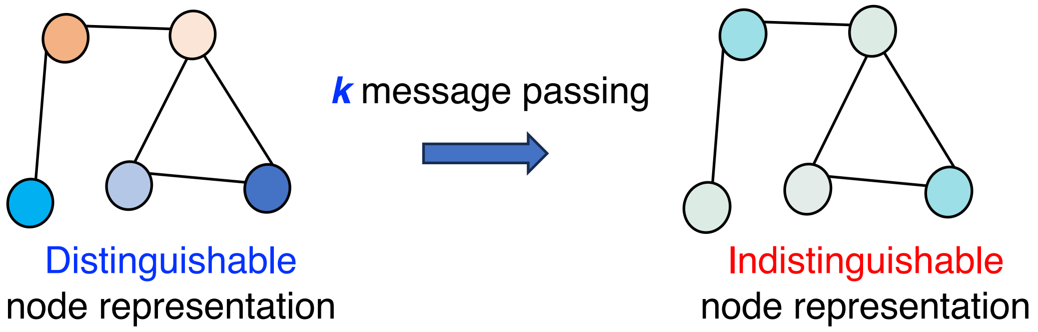
Output: Node representation

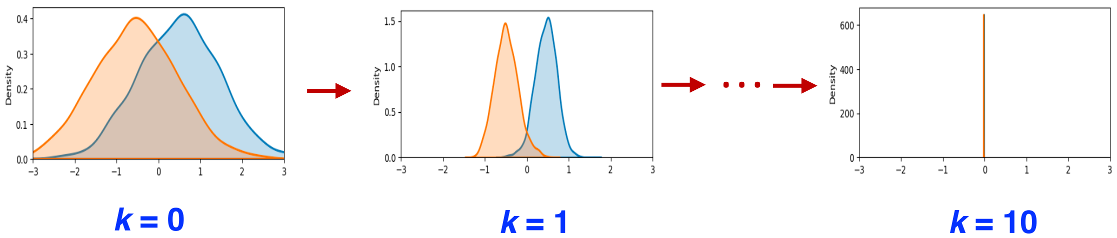


Background: Two limitations of MPNNs

- **Over-smoothing** and **Over-squashing** are two key limitations for developing deep MPNNs

Over-smoothing Issue: As the the number of message passing increases, the node representations **become Indistinguishable**. [Li, AAAI 2018]

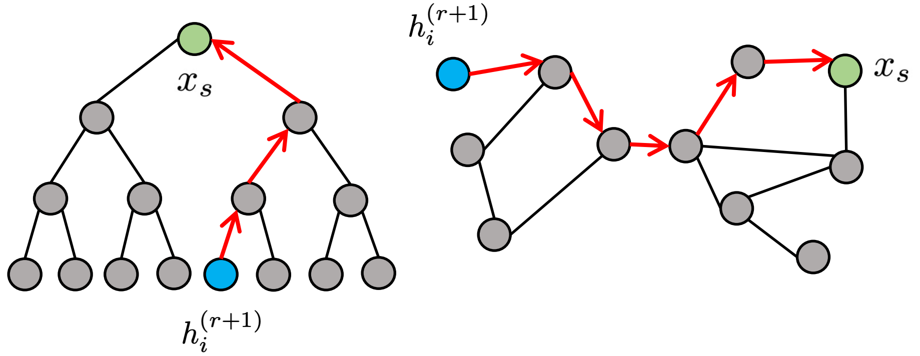




The change of distribution of the representation as the *k* increase

Appropriate smoothing helps with classification, but over-smoothing damages performance. [Keriven, NeurIPS 2022]

Over-squashing Issue: Information from distant nodes gets excessively compressed, **hindering the effective propagation of node features in graph**. [Alon, ICLR 2021]



$$\left|\frac{\partial h_i^{(r+1)}}{\partial x_s}\right| \le (\alpha\beta)^{r+1}(\hat{A}^{r+1})_{is}.$$

[Topping, ICLR2022]

As distance ***r*** increases, $(\hat{A}^{r+1})_{is}$ gives an exponential decay.

Node green representation is insensitive to node blue

Due to over-squashing, the receptive field of GNNs has been greatly restricted. Deep MPNNs lacks effectiveness.

CONTENTS

Related Works: Existing strategies to overcome the above two issues

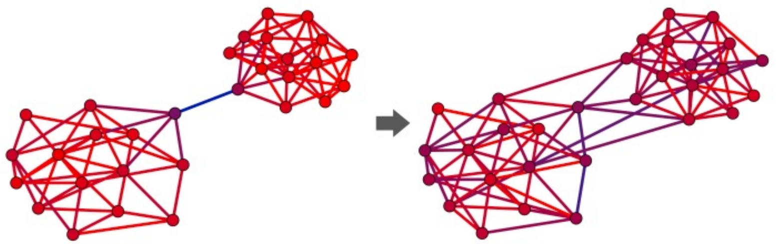
Strategy 1 - Graph Rewiring: Graph rewiring optimizes graph topology through editing topology, to migrate over-smoothing and over-squashing. It can be divided into the following two categories

Rule-based Rewiring

- **DropEdge**[Rong Y, ICLR2020]:
  Random edge removal during training

$$\boldsymbol{A}_{\text{drop}} = \boldsymbol{A} - \boldsymbol{A}',$$

- **SDRF**[Topping, ICLR2021]:
  Edge Addition based on **graph curvature**



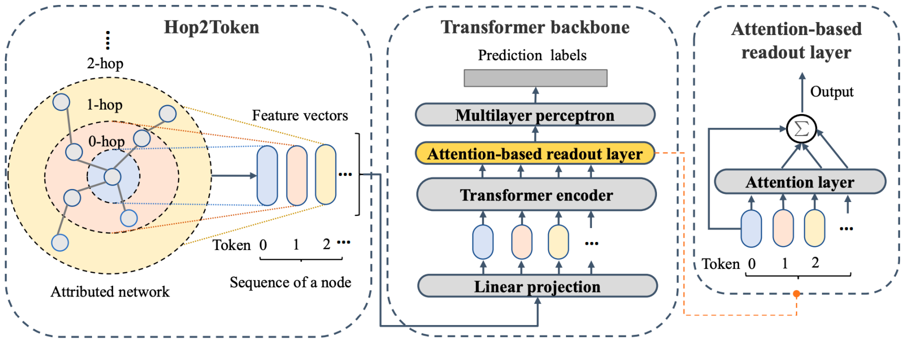
Coloring Edges Based on the Sign of Graph Curvature

Learning-based Rewiring

- **Graphormer**[Ying C, NeruIPS2021]:
  Learning graph structures with **attention mechanism**

$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V,$$
$$A = \frac{QK^\top}{\sqrt{d_K}}, \quad \mathrm{Attn}(H) = \mathrm{softmax}(A)\,V,$$

- **NAGphormer**[Chen j, ICLR2023]:
  using the attention mechanism **rewiring sub-graph**



Main drawback: Disrupt the original graph structure, maybe lead to performance degradation.

Related Works: Existing strategies to overcome the above two issues

Strategy 2 -Regularization : Constraining node representations to be distinctive during training can effect-ively prevent over-smoothing. It can be divided into the following three categories

Constraining Node Representations

EGNN[Zhou, NeurIPS2021]:
Enforcing node similarity through **Dirichlet Energy** constraint

GroupNorm[Zhou, NeurIPS2020]:
Normalize similar **groups of nodes independently**

NodeNorm[Zhou, ICLR2021]:
Normalize node features based on standard deviation to **control the variance of node features**.

Constraining Info Flow

G²[Rusch, ICML2022]:
Dynamic update of node representa-tions based on **gradient adaptation**

GatedGCN[Bresson, Arxiv2018]:
Introduce a **gating mechanism** to control the information flow, learn-ing which edges are more important for down-steam tasks

OPEN[Yang, NeurIPS2022]:
modeling **relevances between pro-pagations** by whole ego-network and multi-channels

Inherent Constraining

ACMP[Wang, ICLR2023]:
Simulating a particle system with **attractive and repulsive forces**, thereby maintaining feature diversity.

GraphCON[Rusch, ICML2023]:
the feature update of each node (oscillator) depends not only on its neighboring nodes but also on the **overall dynamics of the system**.

GRAFF[Giovanni, TMLR2023]:
linear graph convolutions **minimize the Dirichlet energy**

Main drawback: these regularizations may degrade model performance and lack of effective solutions for over-squashing.

Related Works: Existing strategies to overcome the above two issues

Strategy 3 - Residual connection : Fusing shallow GNN representations to migrate over-smoothing or over-squashing. It can be divided into the following two categories:

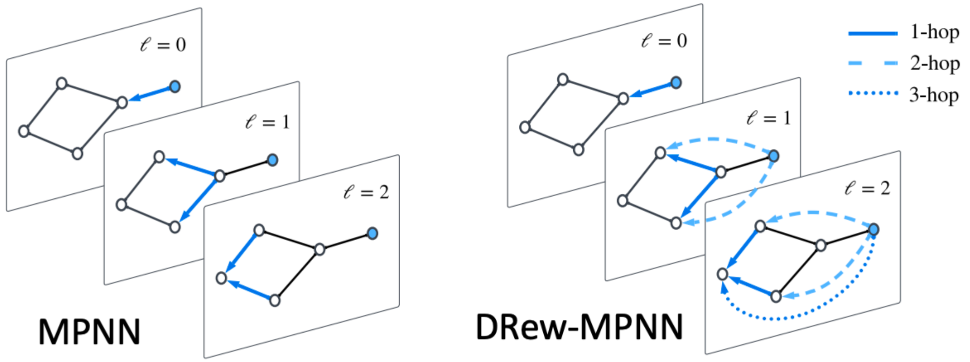
Residual Connections with Initial Features

GCNII[Chen, ICML2020]: Initial residual connections and identity mapping

$$\mathbf{H}^{(\ell+1)}=\sigma\left(\left(\left(1-\alpha_\ell\right)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}+\alpha_\ell\mathbf{H}^{(0)}\right)\left(\left(1-\beta_\ell\right)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}\right)\right)$$

Residual Connections with Shallow Features

Drew[Gutteridge, ICML2023]:

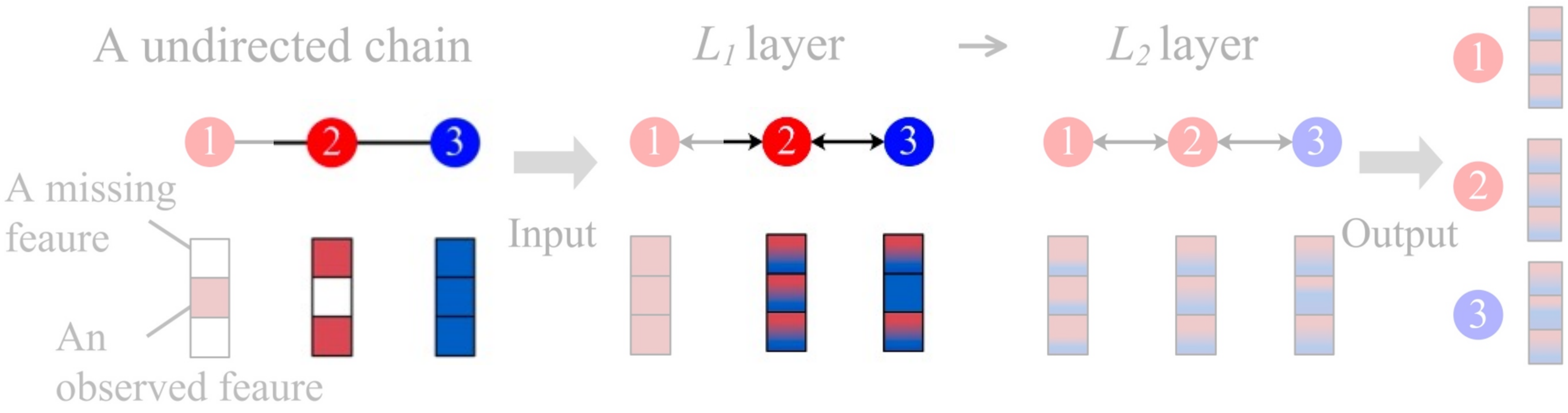


Main drawback: Only alleviates over-smoothing and over-squashing, but doesn't fundamentally resolve.

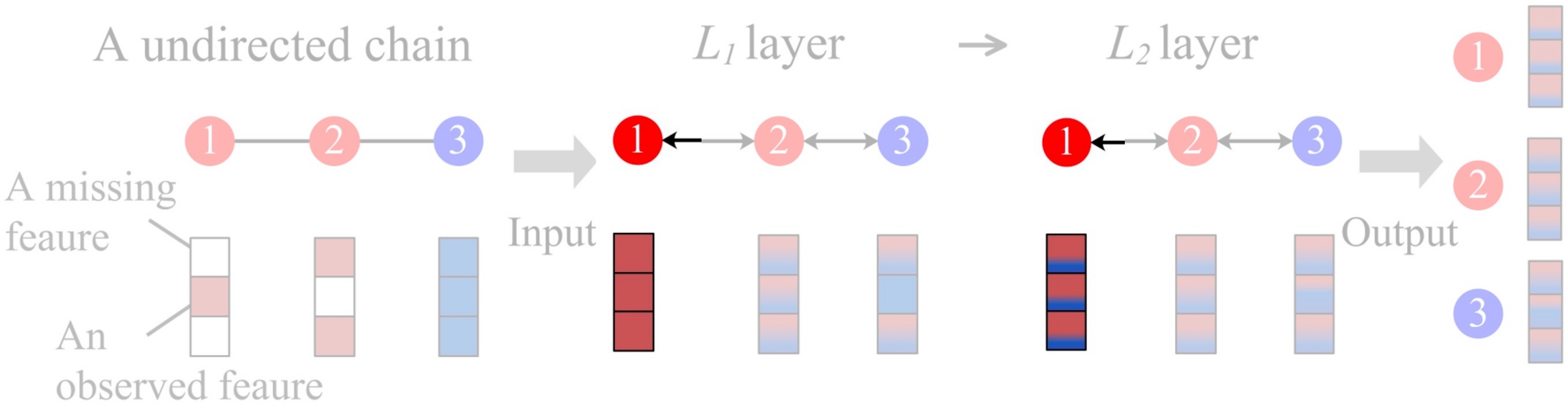
CONTENTS

Method: What is heterophily mixing?

Heterophily mixing is the mixture of messages with **different semantics** (e.g., categories information) **in aggregation of message passing**



The message aggregation of nodes 2 and 3 triggered **heterophily mixing**.

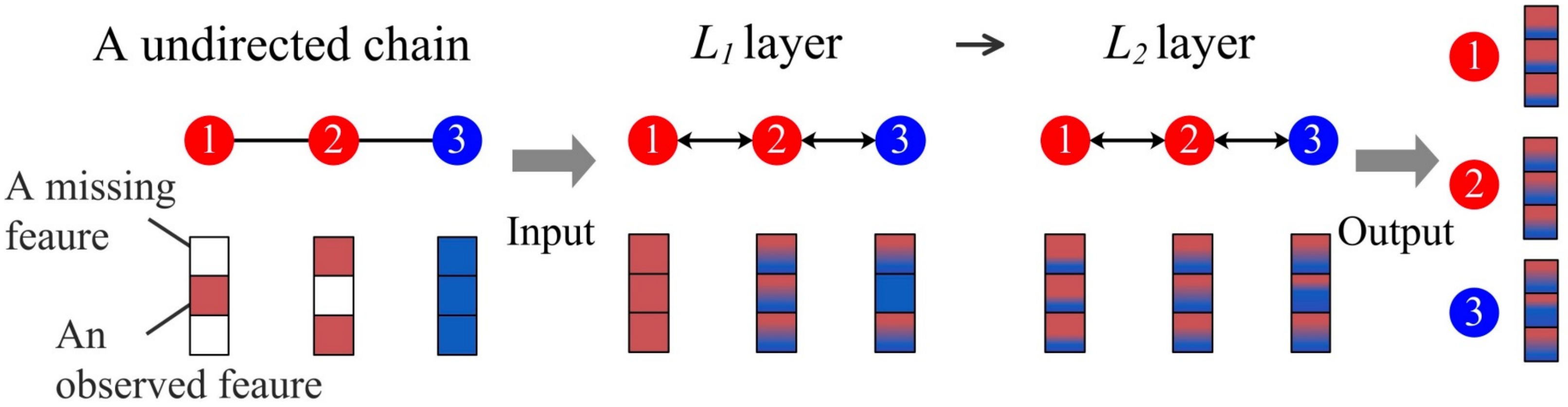


Heterophily mixing can spread as the number of layers increases.

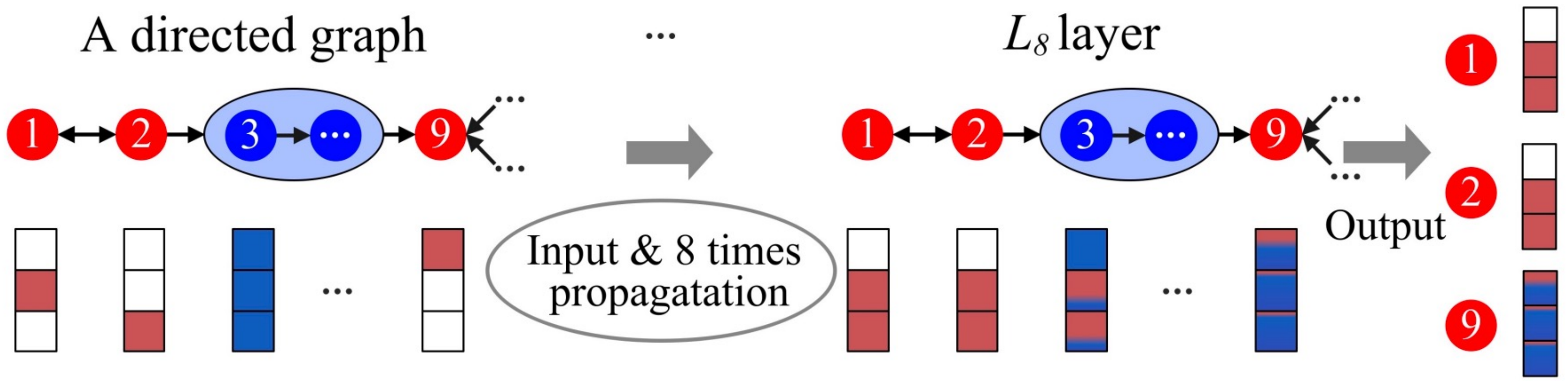
Method: Heterophily mixing restricts the capability of deep MPNNs

Over-smoothing and **over-squashing** are both rooted in information loss resulting from ***heterophily mixing*** in aggregation of message passing

A1. Vanilla *GCN* inherently leads to oversmoothing issue

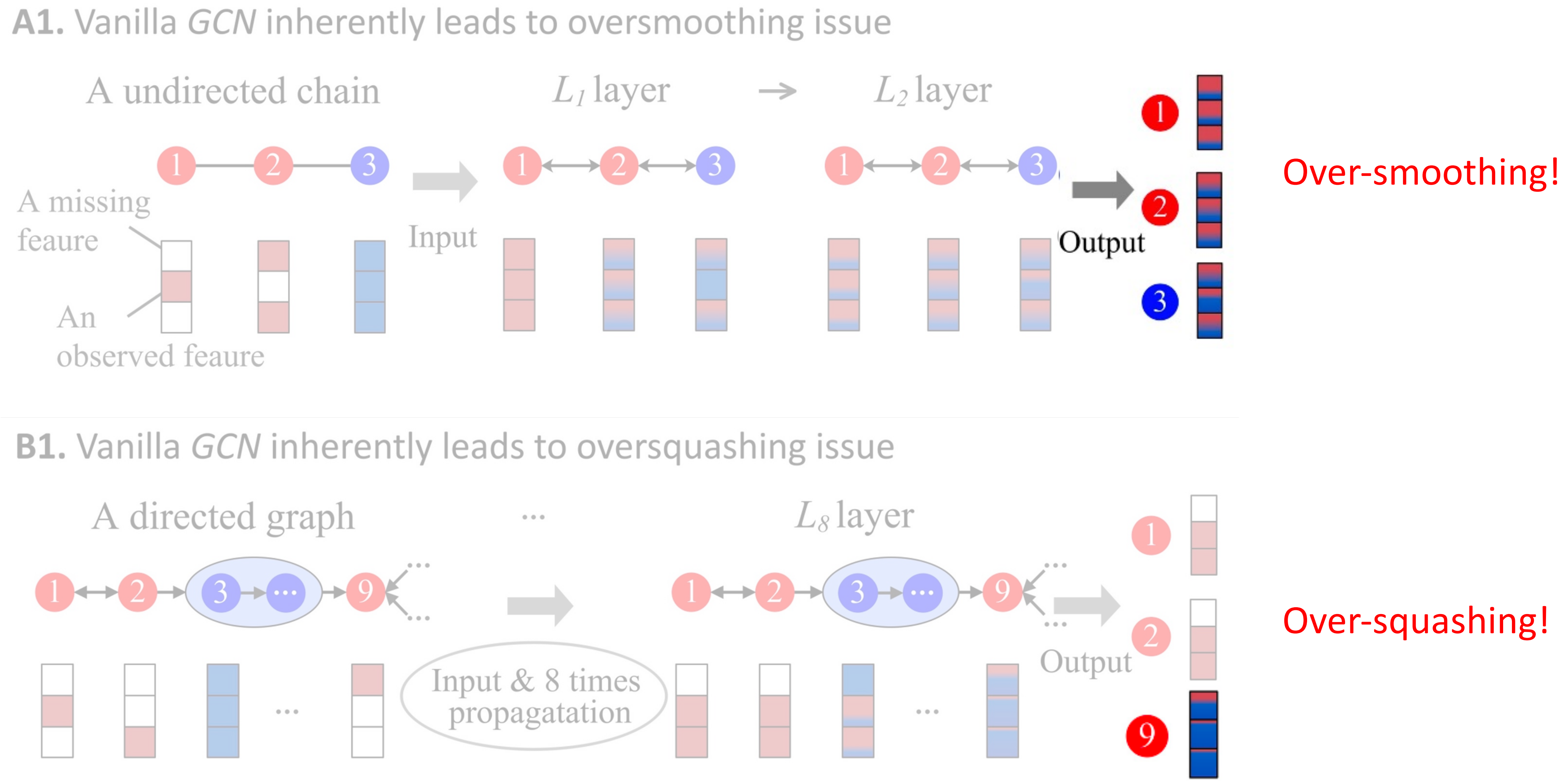


B1. Vanilla *GCN* inherently leads to oversquashing issue



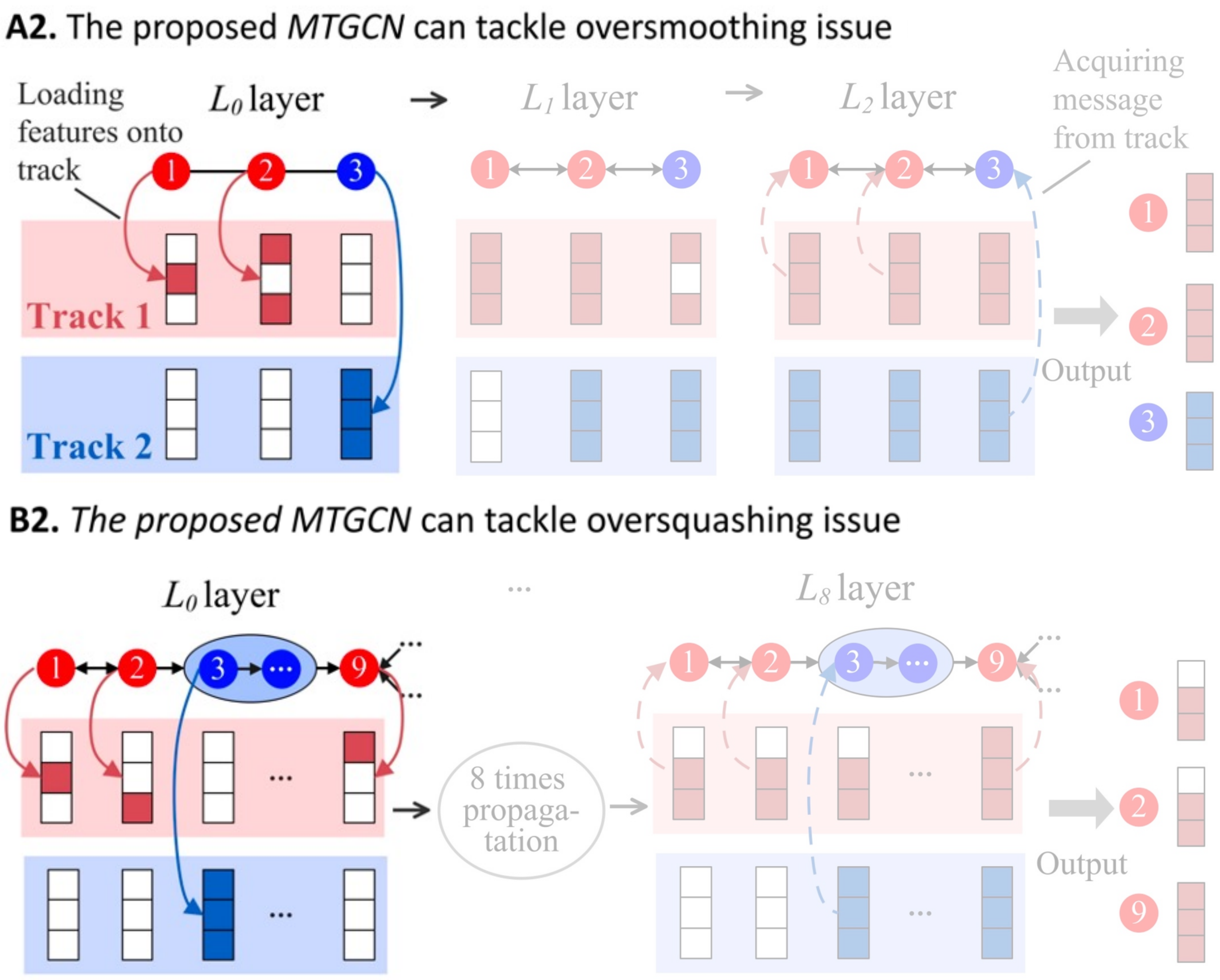
Method: Heterophily mixing restricts the capability of deep MPNNs

Over-smoothing and **over-squashing** are both rooted in information loss resulting from ***heterophily mixing*** in aggregation of message passing



Method: MTGCN Core Intuition

If messages are **separated and independently propagated in tracks** according to their category semantics, heterophilic mixing can be prevented. ⟹ over-smoothing and over-squashing will be addressed effectively



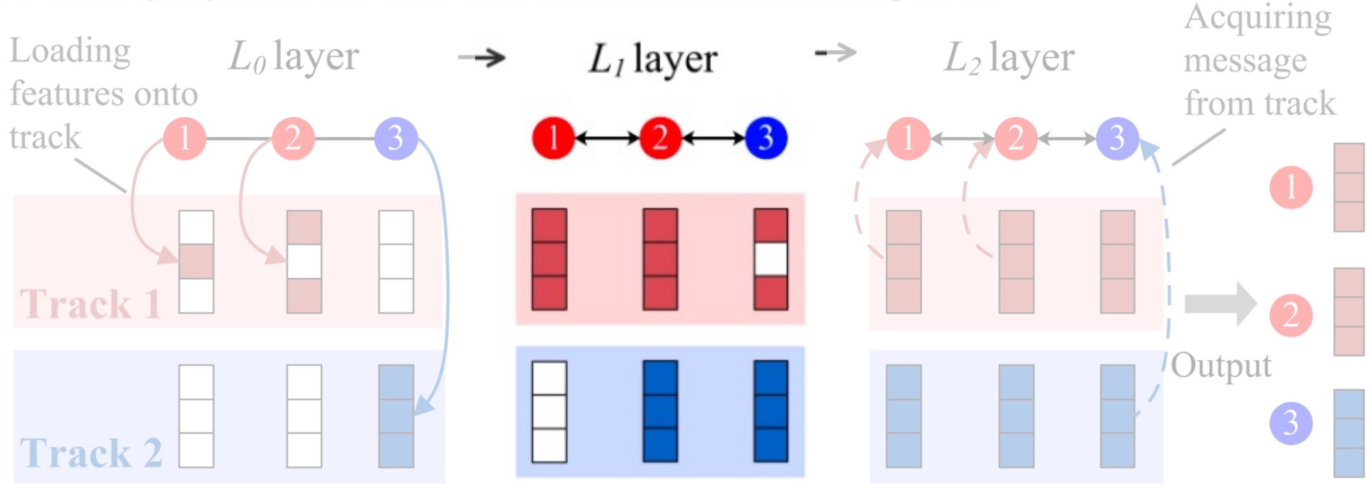
1. Loading

Nodes belonging to the same category are expected to associate with the same track, governed by a node-track affiliation matrix.

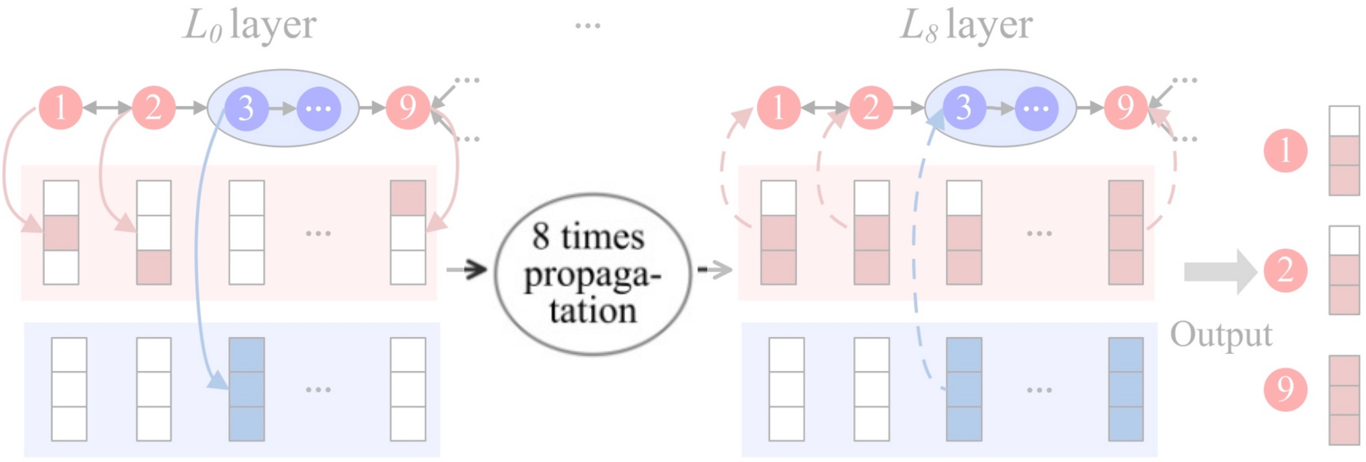
Method: MTGCN Core Intuition

If messages are **separated and independently propagated in tracks** according to their category semantics, heterophilic mixing can be prevented. ⟶ over-smoothing and over-squashing will be addressed effectively

A2. The proposed *MTGCN* can tackle oversmoothing issue



B2. *The proposed MTGCN* can tackle oversquashing issue



1. Loading

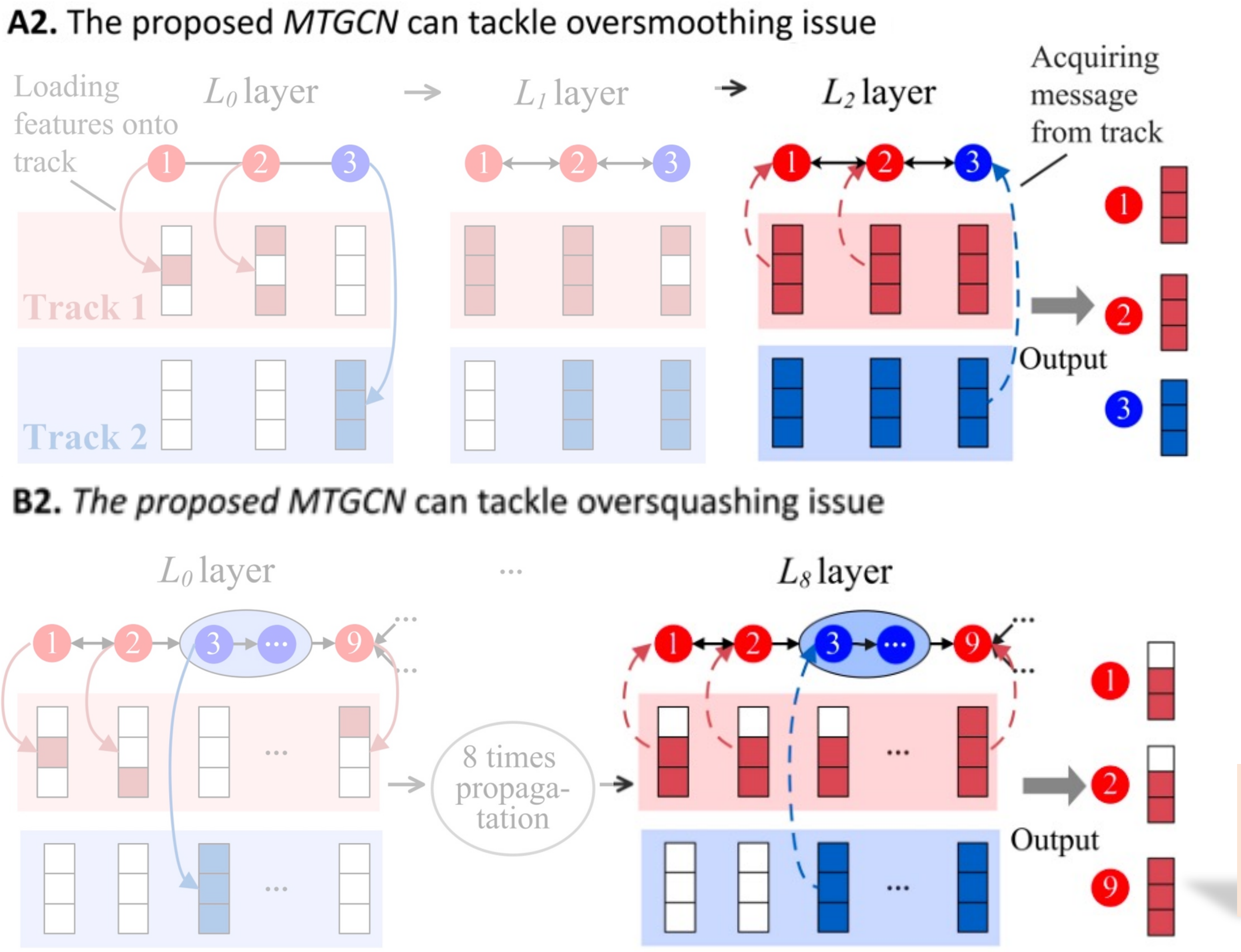
Nodes belonging to the same category are expected to associate with the same track, governed by a node-track affiliation matrix.

2. Multi-Track Message Passing（MTMP）

the initial messages are updated by propagating and aggregating in respective tracks over L iterations.

Method: MTGCN Core Intuition

If messages are **separated and independently propagated in tracks** according to their category semantics, heterophilic mixing can be prevented. ⟹ over-smoothing and over-squashing will be addressed effectively



1. Loading

Nodes belonging to the same category are expected to associate with the same track, governed by a node-track affiliation matrix.

2. Multi-Track Message Passing（MTMP）

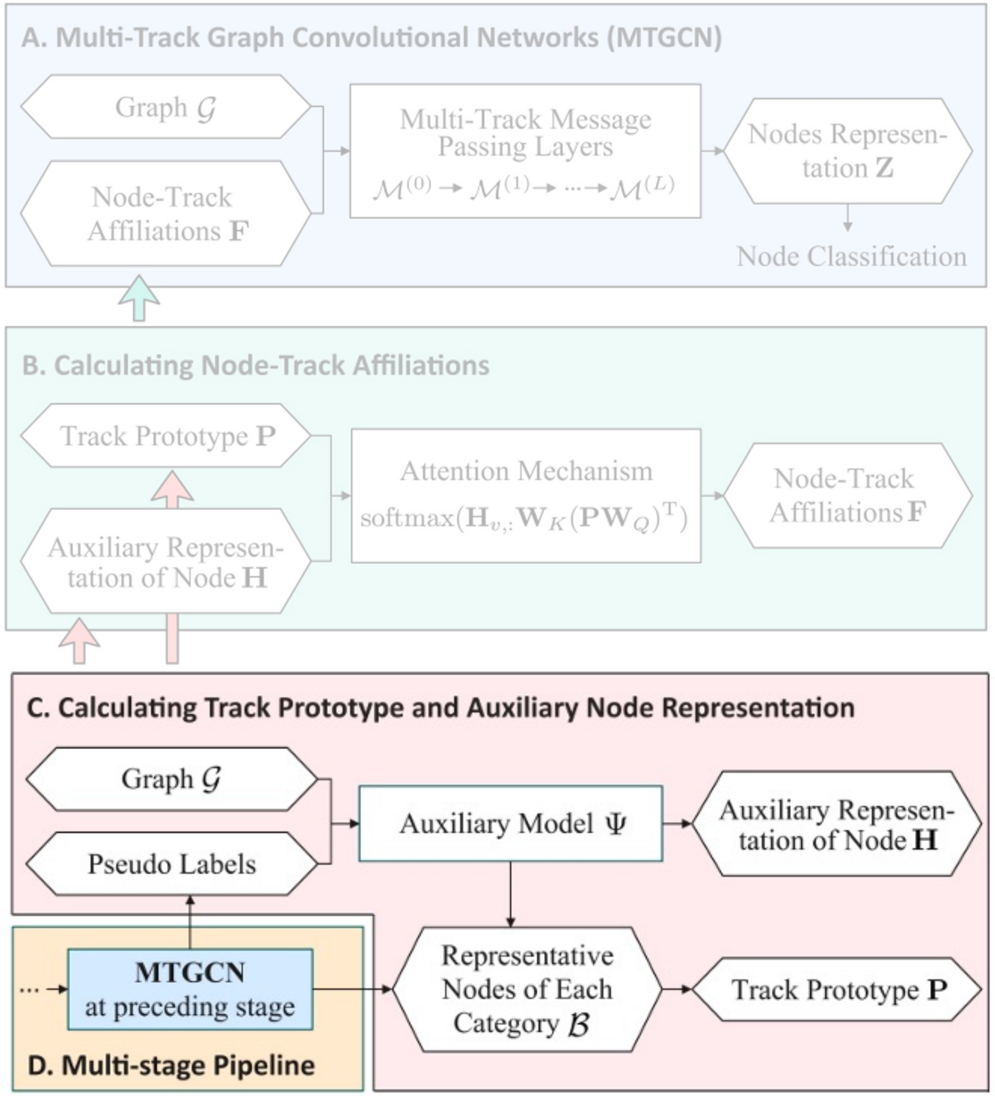
The initial messages are updated by propagating and aggregating in respective tracks over L iterations.

3. Acquiring

Based on the node-track affiliation matrix, nodes acquire the updated messages in their affiliated tracks to construct their node representation.

How to obtain an accurate node-track affiliation matrix?

Method: MTGCN detailed steps



C1: Training auxiliary model

We employ the simple 2-layer GCN[1] as our auxiliary model Ψ

Ψ is trained using both **trainset** and **pseudo labels** (get by prior stage).

C2: calculate track prototype by auxiliary model

$$\mathbf{P}_{T,:} = \frac{1}{\Delta} \sum_{v \in \mathcal{B}} \delta(y_v, T) \cdot \boxed{\mathbf{H}_{v,:}}$$

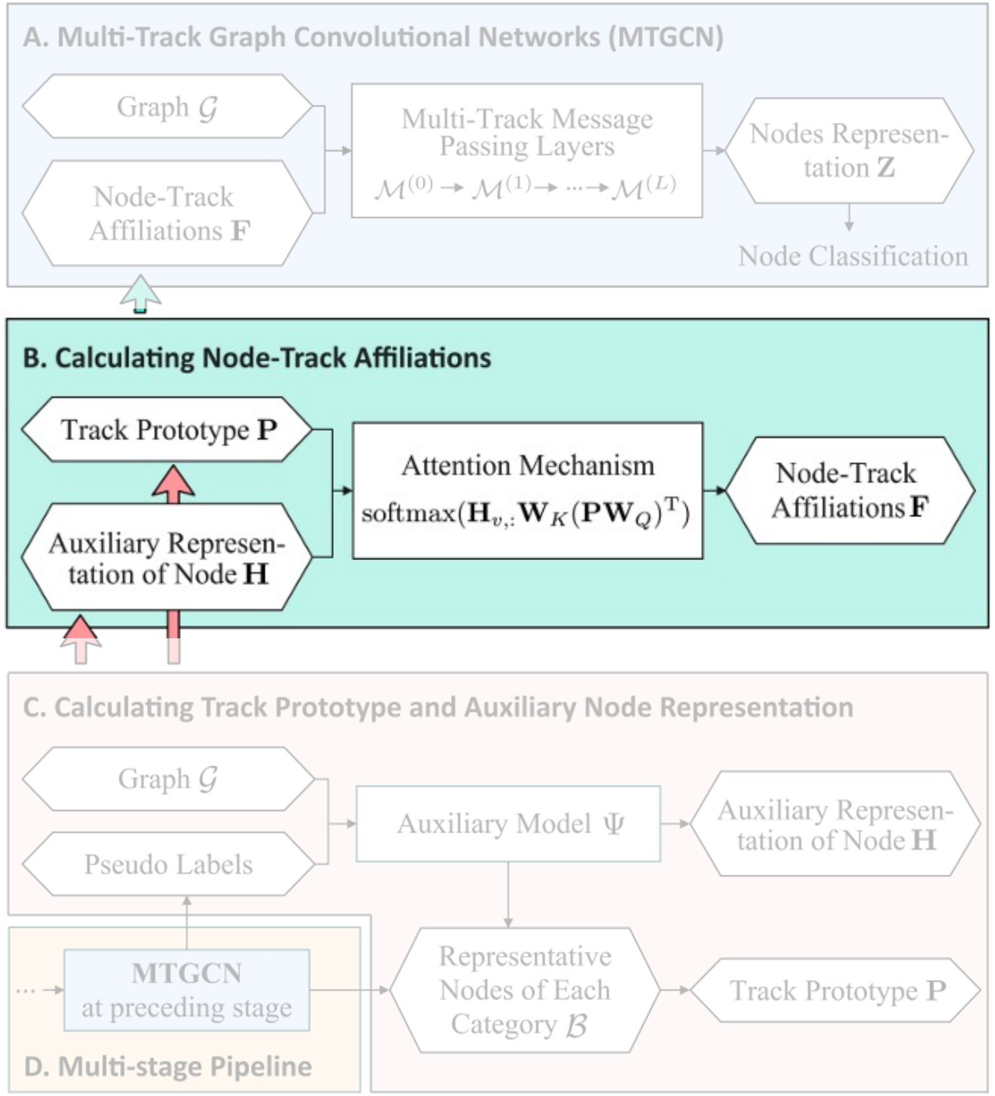
auxiliary model embedding

$\mathcal{B}$ comprises representative node.

$\mathbf{P}$ is the category center of each category of nodes

[1] Kipf, Thomas N., and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." *International Conference on Learning Representations*. 2016.

Method: MTGCN detailed steps



C1: Training auxiliary model

We employ the simple 2-layer GCN[1] as our auxiliary model Ψ

Ψ is trained using both **trainset** and **pseudo labels** (get by prior stage).

C2: calculate track prototype by auxiliary model

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

auxiliary model embedding

$\mathcal{B}$ comprises representative node.

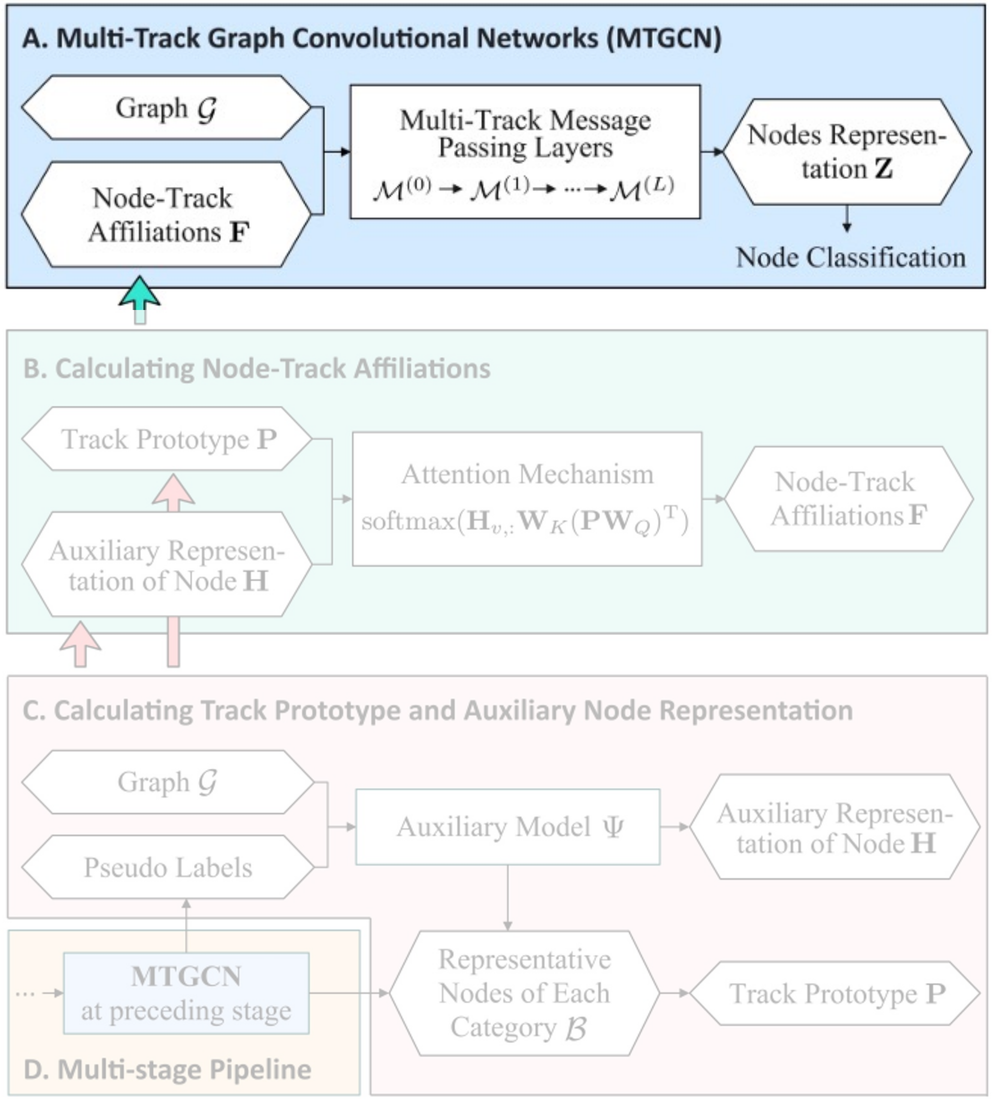
$\mathbf{P}$ is the category center of each category of nodes

B1: Get Node-Track Affiliations

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

[1] Kipf, Thomas N., and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." *International Conference on Learning Representations*. 2016.

Method: MTGCN detailed steps



A1. Loading

$$\boxed{\mathcal{M}^{(0)}_{T,v,:}} = g(\mathbf{X}_{v,:}) \quad \text{if} \quad \boxed{\mathbf{F}_{T,v}} = 1$$
$$\boxed{\mathcal{M}^{(0)}_{T,v,:}} = \vec{0} \quad \text{if} \quad \boxed{\mathbf{F}_{T,v}} = 0,$$

Initial Message

Node-track affiliations

$g: \mathbb{R}^m \to \mathbb{R}^d$ maps node *raw feature* to message space

A2. Multi-Track Message Passing（MTMP）

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

A3. Acquiring

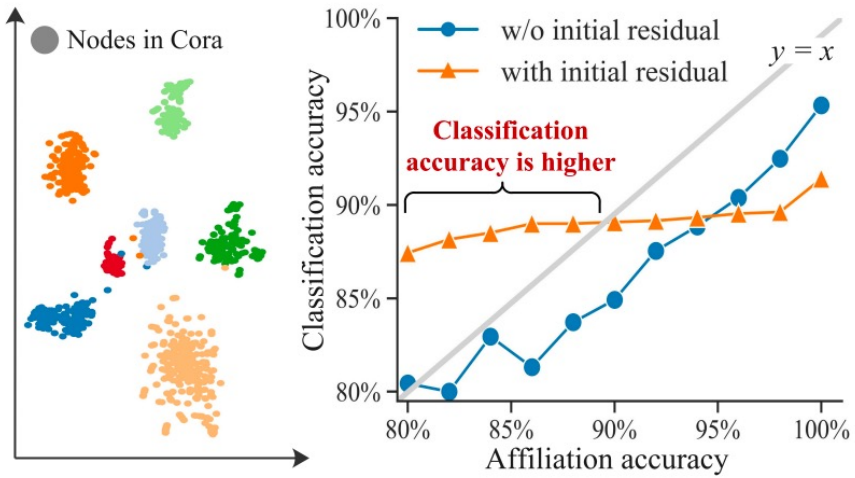
$$\mathbf{Z}_{v,:} = (\sum_{T\in\mathcal{T}} \boxed{\mathbf{F}_{T,v}} \cdot \mathcal{M}^{(L)}_{T,v,:})\boxed{\mathbf{W}_Z},$$

Node-track affiliations

Learnable weight

Method: Why MTMP Gains Improvements?

Capacity 1: Preventing heterophily mixing



A. Node representations **B. Relationship between two accuracies**

Accuracy of $\mathbf{F}_{:,v}$ ⬆ ⟶ heterophily mixing ⬇ ⟶ Classify acc ⬆

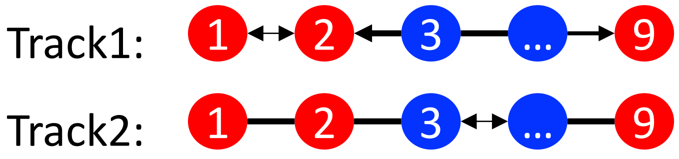
Using **initial residual connections** can achieve higher accuracy when $\mathbf{F}_{:,v}$ low.

Capacity 2: Facilitating long-distant information flow

MPNNs

message passing is inherently tied to the fusion of messages into a node's feature over-squashing

MTMP



It is attributed to the fact that MTMP achieves the **decoupling of messages and node representations.**

Capacity 3: Enhancing separation condition[Wei, ICLR2021]

Expansion assumption:

$$P(\text{Neighborhood}(S)) \geq \min\{c \times P(S), 1\}$$

Separation assumption:

$$E_x[R(G^\star, x)] \leq \mu \text{ for } \mu = \text{small}$$

Different gt classes are sufficiently separated

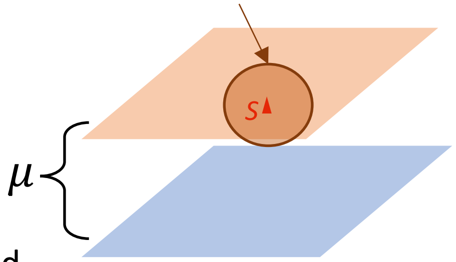


Table 1. Statistics of separation conditions μ

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

$$\text{err}(\hat{G}) \leq \frac{2}{c-1}\underbrace{\text{err}(G_{pl})}_{\text{pseudolabeler}} + \frac{2c}{c-1}\mu$$

⟹ **MTMP can achieve better performance**

CONTENTS

Experiments: node classification task

Semi-supervised node classification

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

- MTGCN demonstrates **superior performance**
- The **multi-stage training strategy** is **highly effective**

Full-supervised node classification

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	Cora	Cite.	Pubm.	Corn.	Texas	Wisc.
Homophily	0.81	0.80	0.74	0.30	0.11	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

- MTGCN is equally **effective on heterogeneous graphs**
- The **multi-stage training strategy** has **limited effectiveness**.

Experiments: solve over-smoothing

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

group distance ratio $R_g = \frac{C}{(C-1)^2} \frac{d_{inter}}{d_{intra}}$

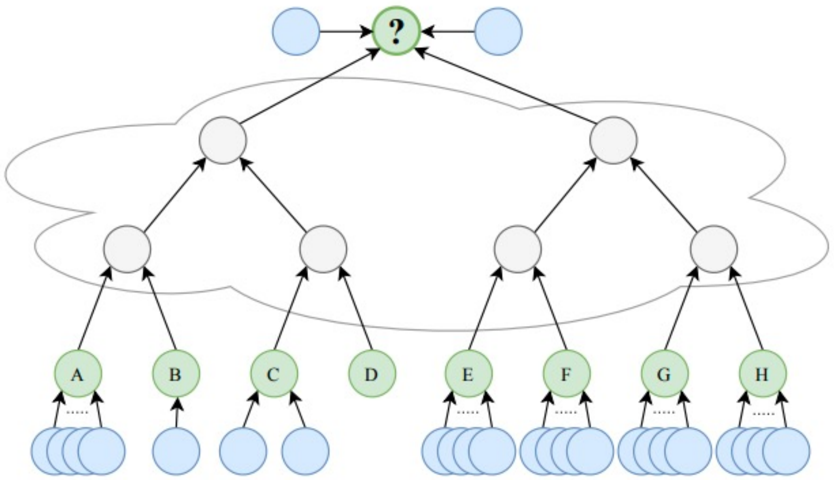
Robustness of MTGCN to depth: MTGCN maintains a stable classification accuracy and group distance ratio (Rg)[1] when increasing the network depth.

"Depth" learning capability of MTGCN: With the increase in the number of layers in MTGCN, its accuracy in classification tasks shows a gradual improvement.

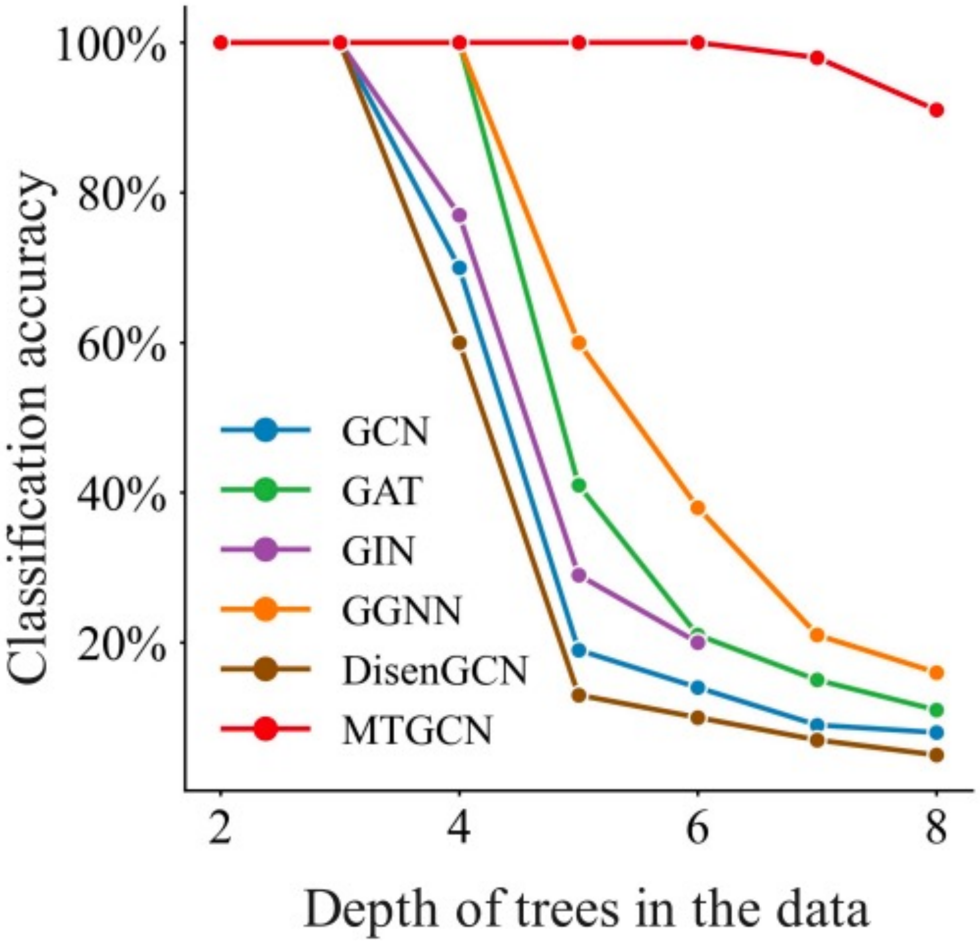
[1]Zhou, Kaixiong, et al. "Towards deeper graph neural networks with differentiable group normalization." *Advances in neural information processing systems* 33 (2020): 4917-4928.

Experiments: Effectively solve over-squashing

Example of the Tree-NeighborsMatch task[1]



Tree-NeighborsMatch Result



MTGCN demonstrates excellent performance： its effectiveness in addressing the over-squashing.

Performance degradation of other models: When the depth of the tree exceeds five layers, the training accuracy of all models, except for MTGCN, significantly decreases.

Slight performance decrease of MTGCN on deep trees: This could be attributed to MTGCN's high spatial complexity.

[1]Alon, Uri, and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications." *International Conference on Learning Representations*. 2020.

CONTENTS

Conclusion

- Heterophilic mixing is one of the key factors leading to over-smoothing and over-squashing.
- A novel Multi-Track Graph Convolutional Network (MTGCN) designed to counteract heterophilic mixing.
- Empirical validation shows that MTGCN performs well and solves the problems of over-smoothing and over-squashing.

Paper: https://openreview.net/pdf?id=1sRuv4cnuZ

Code： https://github.com/XJTU-Graph-Intelligence-Lab/mtgcn

If you have any problems, please feel free to contact me:
liyu1998@stu.xjtu.edu.cn, peihongbin@xjtu.edu.cn