MGNN: Graph Neural Networks Inspired by Distance Geometry Problem

Guanyu Cui [1] Zhewei Wei* [1]

[1]Renmin University of China

Graph Neural Networks (GNNs)

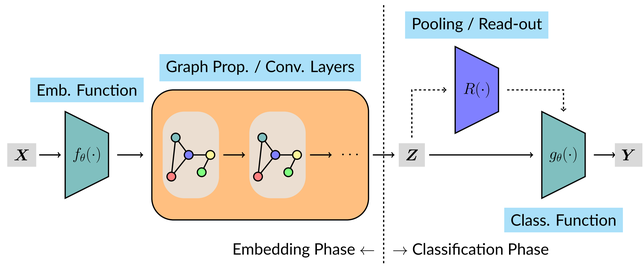


Figure 1. A typical structure of GNNs.

- Embedding function $f(\boldsymbol{X})$: map features $\boldsymbol{X} \in \mathbb{R}^{n\times F}$ to initial embeddings $\boldsymbol{Z}^{(0)} \in \mathbb{R}^{n\times d}$;
- Graph propagation / convolution $\mathrm{GP}(\boldsymbol{Z}^{(0)}; \boldsymbol{A})$: propagate embedding K times;
- Read-out / pooling $\mathrm{R}(\boldsymbol{Z}^{(K)}; \boldsymbol{A})$ (optional): pooling for graph-level tasks;
- Classification function $g(\cdot)$: final classification to generate predictions $\boldsymbol{Y}$.

Expressive Power & Universality of GNNs

- Spectral GNNs: designing **universal filters** (e.g., ChebNet, GPRGNN, BernNet, ...).
- Spatial GNNs: designing GNNs bounded by k**-WL tests** (e.g., GIN, ...).

There are works exploring relations between GNNs and geometric objects (e.g., curvature, cellular sheaves) and physical concepts (e.g., oscillators).

Q: Can we define universality of spatial GNNs from a geometric perspective?

Preliminaries

Definition (Equivalent). For a given graph $G = (V, E)$, two node embedding matrices $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)} \in \mathbb{R}^{n\times d}$ are equivalent if for all $(i, j) \in E$, $\|\boldsymbol{Z}^{(1)}_{i:} - \boldsymbol{Z}^{(1)}_{j:}\|_2 = \|\boldsymbol{Z}^{(2)}_{i:} - \boldsymbol{Z}^{(2)}_{j:}\|_2$ holds.

Definition (Congruent). For a given graph $G = (V, E)$, two node embedding matrices $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)} \in \mathbb{R}^{n\times d}$ are congruent if for all $i, j \in V$, $\|\boldsymbol{Z}^{(1)}_{i:} - \boldsymbol{Z}^{(1)}_{j:}\|_2 = \|\boldsymbol{Z}^{(2)}_{i:} - \boldsymbol{Z}^{(2)}_{j:}\|_2$ holds.

Definition (Globally Rigid). For a given graph $G = (V, E)$, an embedding matrix $\boldsymbol{Z}$ is globally rigid if all its equivalent embedding matrices $\boldsymbol{Z}'$ are also congruent to $\boldsymbol{Z}$.

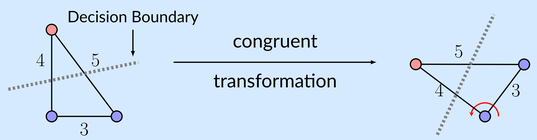
Definition (Rigid). For a given graph $G = (V, E)$, an embedding matrix $\boldsymbol{Z}$ is rigid if all equivalent embeddings that can be obtained by **continuous** motion from $\boldsymbol{Z}$ are congruent to $\boldsymbol{Z}$.

Figure 2. Left: A globally rigid graph in $\mathbb{R}^2$; Right: A rigid but not globally rigid graph in $\mathbb{R}^2$, since it has an equivalent but not congruent embedding.

Definition (Metric Matrix). The metric matrix of an embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$ is defined as $\boldsymbol{M}_{\boldsymbol{Z}} = (\|\boldsymbol{Z}_{i:} - \boldsymbol{Z}_{j:}\|_2)_{ij}$. We also define the mapping from an embedding matrix $\boldsymbol{Z}$ to its metric matrix $\boldsymbol{M}_{\boldsymbol{Z}}$ as $\boldsymbol{M}_{\boldsymbol{Z}} = M(\boldsymbol{Z})$.

Defining Spatial-Universality

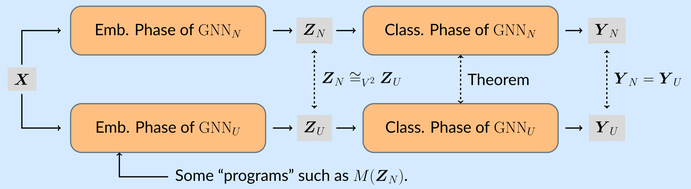
Observation:



Defining Spatial-Universality (Cont'd)

Theorem (MLPs Are Congruent-Insensitive). Given two congruent embedding matrices $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$, for any MLP_M (with biases), there always exists another MLP_N (also with biases) such that $\mathrm{MLP}_M(\boldsymbol{Z}_1) = \mathrm{MLP}_N(\boldsymbol{Z}_2)$.

Idea:



Spatial-Universal: It can arrange nodes with a given metric matrix!

This idea is closely related to the **Distance Geometry Problem (DGP)**.

Distance Geometry Problem (DGP)

Given a positive integer d, a graph $G = (V, E)$, and a symmetric non-negative matrix $\boldsymbol{M}$, decide whether there exists an embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$, such that

$$\forall (i, j) \in E, \|\boldsymbol{Z}_{i:} - \boldsymbol{Z}_{j:}\| = \boldsymbol{M}_{ij}.$$

Optimization Objective

About the Optimization Objective

- **Full** metric matrix: $O(n^2) \Rightarrow$ **partial** metric matrix on edges: $O(m)$.
- For **globally rigid** graphs, partial metric matrix is enough to determine the "shape" of the embedding, this modification does not weaken the expressive power.
- However, solving the DGP is **NP-Hard**. We can not directly arrange the nodes, thus we introduce an **error-tolerant objective**:

$$E_p(\boldsymbol{Z}; \boldsymbol{M}, E) = \frac{1}{2}\|\boldsymbol{A} \odot (M(\boldsymbol{Z}) - \boldsymbol{M})\|_F^2 = \sum_{(i,j)\in E} \frac{1}{2}\left(\|\boldsymbol{Z}_{i:} - \boldsymbol{Z}_{j:}\|_2 - \boldsymbol{M}_{ij}\right)^2.$$

- This function is closely related to the raw **Stress function** σ_r in the **Multidimensional Scaling (MDS)** problem and the potential energy of **spring networks**.

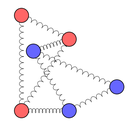


Figure 3. A spring network.

- To align with other representative GNNs, we modify E_p and add a regularization term to get the final objective:

$$\begin{aligned}\mathcal{L}(\boldsymbol{Z}; \boldsymbol{Z}^{(0)}, \boldsymbol{M}, E) &= (1-\alpha)\tilde{E}_p(\boldsymbol{Z}; \boldsymbol{M}, E) + \alpha\|\boldsymbol{Z} - \boldsymbol{Z}^{(0)}\|_F^2 \\ &= (1-\alpha)E_p(\boldsymbol{D}^{1/2}\boldsymbol{Z}; \boldsymbol{M}, E) + \alpha\|\boldsymbol{Z} - \boldsymbol{Z}^{(0)}\|_F^2.\end{aligned}$$

About the Metric Matrix

- For scenarios with prior knowledge about distances between nodes (e.g., molecular conformation generation, or graph drawing), directly use them; for other scenarios, learn a metric matrix.
- Our idea is to increase the distances between dissimilar nodes and reduce the distances between similar nodes:
  1. Introduce edge attention $\alpha_{ij} \in [-1, 1]$, when $\alpha_{ij} \to 1 \Leftrightarrow i, j$ tend to belong to the same class, and when $\alpha_{ij} \to -1 \Leftrightarrow i, j$ tend to belong to different classes;
  2. Map the initial embedding matrix $\boldsymbol{Z}^{(0)}$ (defined later) to a hidden matrix $\boldsymbol{H}$;
  3. Use attention mechanisms, such as $\alpha_{ij} = \tanh\left(\boldsymbol{a}^\top[\boldsymbol{H}_{i:}^\top \| \boldsymbol{H}_{j:}^\top]\right)$ or $\alpha_{ij} = \tanh\left(\boldsymbol{H}_{i:}\boldsymbol{W}\boldsymbol{H}_{j:}^\top\right)$ to learn the edge attention;
  4. Then we can set $\boldsymbol{M}_{ij} = \frac{1-\alpha_{ij}}{1+\alpha_{ij}+\varepsilon}\|\boldsymbol{Z}^{(0)}_{i:} - \boldsymbol{Z}^{(0)}_{j:}\|$, where ε is a small positive number.

Framework

The Embedding Function

- A linear layer $f(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{1}\boldsymbol{b}^\top$ in linear GNNs, or
- A two-layer MLP $f(\boldsymbol{X}) = \sigma(\sigma(\boldsymbol{X}\boldsymbol{W}_1 + \boldsymbol{1}\boldsymbol{b}_1^\top)\boldsymbol{W}_2 + \boldsymbol{1}\boldsymbol{b}_2^\top)$ in spectral GNNs.

Propagation

- Our goal is to design a propagation method that minimizes the objective.
- It's non-convex. Following related works, we employ the **stationary point iteration** method.
- By computing the gradient, setting it to zero, rearranging the terms, rewriting it as an iteration form, substituting $1-\alpha$ with β to allow more flexibility, it leads to:

$$\boldsymbol{Z}^{(k+1)} = (1-\alpha)\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\boldsymbol{Z}^{(k)} + \beta\boldsymbol{D}^{-1/2}\boldsymbol{L}_{\boldsymbol{H}}\boldsymbol{D}^{-1/2}\boldsymbol{Z}^{(k)} + \alpha\boldsymbol{Z}^{(0)},$$

  where $\boldsymbol{H} = \boldsymbol{A} \odot \boldsymbol{M} \odot M(\boldsymbol{D}^{-1/2}\boldsymbol{Z})^{\odot -1}$, and $\boldsymbol{L}_{\boldsymbol{H}} = \mathrm{diag}(\boldsymbol{H}\boldsymbol{1}) - \boldsymbol{H}$.
- And we have the message-passing form:

$$\boldsymbol{Z}^{(k+1)}_{i:} = (1-\alpha)\sum_{j\in\mathcal{N}(i)} \frac{\boldsymbol{Z}^{(k)}_{i:}}{\sqrt{d_i d_j}} + \beta\sum_{j\in\mathcal{N}(i)} \frac{\boldsymbol{M}_{ij}\left(\boldsymbol{Z}^{(k)}_{i:} - \boldsymbol{Z}^{(k)}_{j:}\right)}{\sqrt{d_i d_j}\left\|\boldsymbol{Z}^{(k)}_{i:}/\sqrt{d_i} - \boldsymbol{Z}^{(k)}_{j:}/\sqrt{d_j}\right\|_2} + \alpha\boldsymbol{Z}^{(0)}_{i:}.$$

Optional Linear and Non-linear Transformations

We may incorporate linear and non-linear transformations after each propagation step. In our experiments without pre-designed metric matrices, such as node classification, we utilize three designs from the GCNII model: a linear transformation, the identity mapping, and a non-linear transformation (ReLU).

The Classification Function

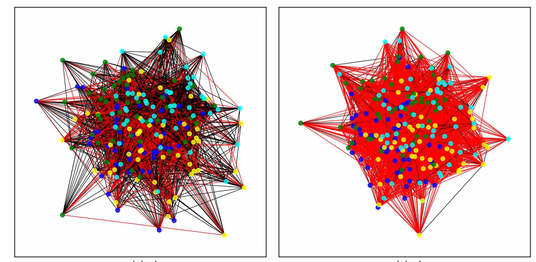
We use a linear layer $g(\boldsymbol{Z}^{(K)}) = \boldsymbol{Z}^{(K)}\boldsymbol{W} + \boldsymbol{1}\boldsymbol{b}^\top$ as the final classification function.

Experiments

We have done the "Arranging Nodes with Given Metric Matrices" experiments on synthetic graphs, supervised node classification and graph regression experiments on real-world graphs.

Arranging Nodes with Given Metric Matrices

- We generate two **Stochastic Block Model (SBM)** graphs, one homophilic and one heterophilic, consisting of four blocks with 50 nodes in each block.
- The node features are sampled from two 2-dimensional Gaussian distributions.



- If i and j are in the same class, we set $\boldsymbol{M}_{ij} = 0$; otherwise, we set $\boldsymbol{M}_{ij} = 5$.
- We pass the node features through 8 MGNN layers, with $\alpha = 0.05$ and $\beta = 0.5$.
- For visualization results, please refer to the smaller posters below or our paper.

Supervised Node Classification and Graph Regression

- Our MGNN model performs well, and the results are promising.
- For experiment details and results, please refer to the smaller posters below or our paper.

*Zhewei Wei is the corresponding author. Contact us: cuiguanyu@ruc.edu.cn; zhewei@ruc.edu.cn

MGNN: Graph Neural Networks Inspired by Distance Geometry Problem

Guanyu Cui [1] Zhewei Wei* [1]

[1]Renmin University of China

Experiment I: Arranging Nodes with Given Metric Matrices

- We pass the node features through 8 MGNN layers, with $\alpha = 0.05$ and $\beta = 0.5$. After each round of propagation, we visualize the results.

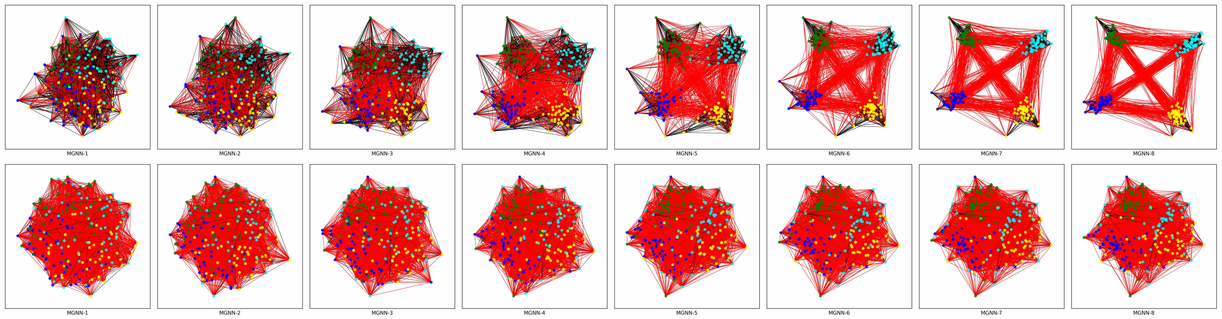


Figure 1. Visualization results of MGNN propagation layers.

- Based on the results, we can confirm that our MGNN model aims to separate the four blocks.
- The presence of homophilic edges adds convexity to our objective function, which is easier to optimize.
- Additionally, we provide visualization results obtained from the SGC layers and APPNP layers.

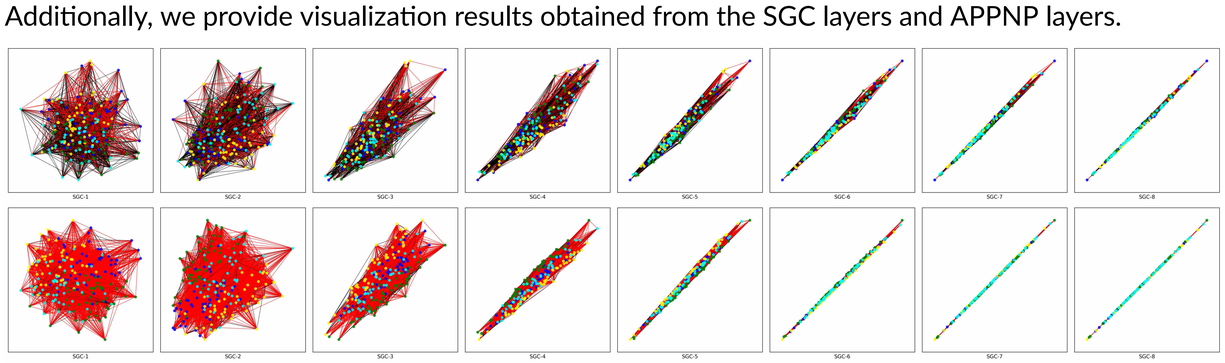


Figure 2. Visualization results of SGC propagation layers.

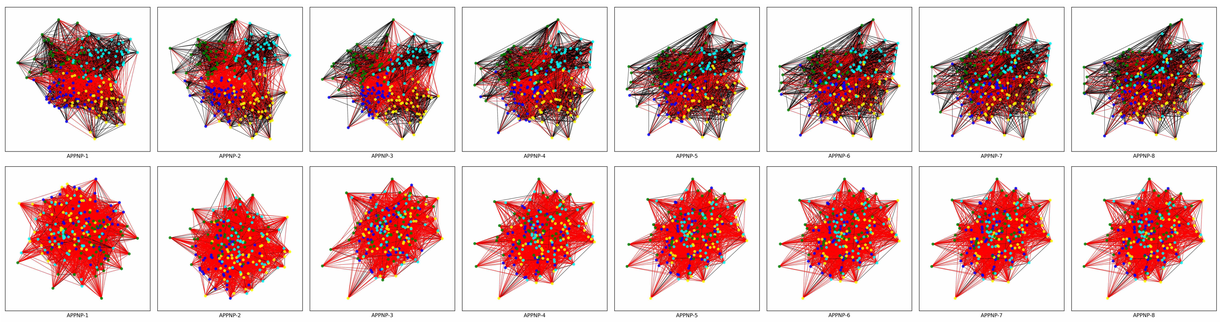


Figure 3. Visualization results of APPNP propagation layers.

Experiment II: Supervised Node Classification

- For all datasets except ogbn-arxiv, we generate 10 random splits with train/valid/test ratio of 60%/20%/20%. For the ogbn-arxiv dataset, we generate 10 random splits using the `get_idx_split()` function provided by the official package.
- We then tune the hyper-parameters 100 times for each model on each dataset. (Please refer to our paper for the detailed ranges of hyperparameters.)
- We train each model for a maximum of 1500 epochs, with early stopping set to 100, on each dataset and report the results after hyper-tuning.

Table 1. The results (accuracy with standard deviation) of the supervised node classification experiments. Boldface results indicate the best model on each dataset, and underlined results are second best models.

	Non-Graph		Spectral			Spatial (Non-Spectral)						MGNN
	Linear	MLP	GCN	SGC	APPNP	PointNet	GAT	GIN	GCNII	^{p}GNN	LINKX	
Cora	76.09$_{\pm 1.55}$	76.11$_{\pm 1.58}$	88.25$_{\pm 1.09}$	88.34$_{\pm 1.41}$	**89.30$_{\pm 1.45}$**	84.43$_{\pm 1.94}$	88.71$_{\pm 0.89}$	85.76$_{\pm 1.18}$	88.52$_{\pm 1.40}$	88.78$_{\pm 1.20}$	82.82$_{\pm 1.96}$	89.04$_{\pm 1.20}$
CiteSeer	71.11$_{\pm 1.81}$	73.25$_{\pm 1.16}$	77.10$_{\pm 1.12}$	77.47$_{\pm 1.42}$	76.80$_{\pm 1.10}$	72.83$_{\pm 1.38}$	76.44$_{\pm 1.19}$	72.83$_{\pm 1.47}$	77.05$_{\pm 0.78}$	77.28$_{\pm 0.88}$	71.79$_{\pm 1.55}$	**78.08$_{\pm 1.50}$**
PubMed	87.06$_{\pm 0.67}$	88.21$_{\pm 0.46}$	89.07$_{\pm 0.42}$	87.59$_{\pm 0.54}$	89.75$_{\pm 0.47}$	89.18$_{\pm 0.38}$	87.77$_{\pm 0.67}$	87.15$_{\pm 0.54}$	90.24$_{\pm 0.55}$	89.79$_{\pm 0.38}$	86.77$_{\pm 0.59}$	**90.37$_{\pm 0.47}$**
CoraFull	60.55$_{\pm 0.76}$	61.61$_{\pm 0.56}$	71.86$_{\pm 0.75}$	71.86$_{\pm 0.70}$	71.88$_{\pm 0.75}$	63.32$_{\pm 1.02}$	70.65$_{\pm 0.86}$	67.11$_{\pm 0.46}$	71.09$_{\pm 0.72}$	72.36$_{\pm 0.58}$	63.57$_{\pm 0.55}$	**72.42$_{\pm 0.69}$**
CS	94.51$_{\pm 0.32}$	94.89$_{\pm 0.23}$	93.77$_{\pm 0.37}$	94.10$_{\pm 0.40}$	95.91$_{\pm 0.23}$	93.13$_{\pm 0.42}$	93.25$_{\pm 0.31}$	91.81$_{\pm 0.34}$	95.98$_{\pm 0.22}$	95.83$_{\pm 0.23}$	95.06$_{\pm 0.24}$	**96.01$_{\pm 0.16}$**
Physics	95.92$_{\pm 0.15}$	96.01$_{\pm 0.26}$	96.46$_{\pm 0.25}$	OOM	97.14$_{\pm 0.21}$	96.37$_{\pm 0.27}$	96.47$_{\pm 0.21}$	94.66$_{\pm 2.04}$	97.10$_{\pm 0.21}$	96.93$_{\pm 0.10}$	96.87$_{\pm 0.16}$	**97.15$_{\pm 0.15}$**
Cornell	78.65$_{\pm 2.82}$	75.95$_{\pm 4.43}$	50.27$_{\pm 6.86}$	53.51$_{\pm 4.32}$	77.84$_{\pm 6.49}$	71.08$_{\pm 5.93}$	50.27$_{\pm 7.66}$	49.46$_{\pm 7.46}$	76.76$_{\pm 7.76}$	77.30$_{\pm 8.22}$	71.08$_{\pm 12.09}$	**81.89$_{\pm 6.29}$**
Texas	81.35$_{\pm 4.90}$	83.78$_{\pm 7.55}$	61.08$_{\pm 8.65}$	56.22$_{\pm 6.37}$	86.76$_{\pm 1.46}$	82.16$_{\pm 6.30}$	61.08$_{\pm 5.57}$	64.05$_{\pm 4.20}$	88.65$_{\pm 4.95}$	85.14$_{\pm 5.16}$	84.32$_{\pm 5.24}$	**90.00$_{\pm 2.72}$**
Wisconsin	86.20$_{\pm 3.63}$	**88.80$_{\pm 2.40}$**	55.80$_{\pm 5.83}$	58.00$_{\pm 4.29}$	86.00$_{\pm 3.10}$	81.60$_{\pm 3.98}$	56.40$_{\pm 5.99}$	57.20$_{\pm 6.82}$	88.20$_{\pm 4.14}$	85.40$_{\pm 3.58}$	82.00$_{\pm 3.10}$	88.40$_{\pm 3.44}$
Chameleon	50.77$_{\pm 2.07}$	50.75$_{\pm 2.13}$	69.21$_{\pm 2.08}$	67.12$_{\pm 2.17}$	67.85$_{\pm 2.65}$	63.76$_{\pm 2.52}$	66.97$_{\pm 2.45}$	46.73$_{\pm 13.51}$	67.56$_{\pm 1.18}$	69.19$_{\pm 1.57}$	67.47$_{\pm 1.62}$	**72.37$_{\pm 2.25}$**
Squirrel	35.76$_{\pm 1.05}$	35.79$_{\pm 1.65}$	55.43$_{\pm 2.05}$	52.18$_{\pm 1.49}$	54.60$_{\pm 1.88}$	47.39$_{\pm 8.31}$	55.68$_{\pm 2.81}$	20.96$_{\pm 2.08}$	53.88$_{\pm 2.77}$	51.61$_{\pm 1.28}$	**57.86$_{\pm 1.17}$**	54.45$_{\pm 1.85}$
Actor	36.16$_{\pm 0.75}$	37.67$_{\pm 1.60}$	30.42$_{\pm 1.58}$	30.14$_{\pm 1.18}$	36.98$_{\pm 1.28}$	36.41$_{\pm 1.23}$	29.22$_{\pm 0.94}$	26.09$_{\pm 1.75}$	37.75$_{\pm 1.23}$	36.47$_{\pm 1.07}$	35.00$_{\pm 2.11}$	**38.36$_{\pm 1.46}$**
WikiCS	78.74$_{\pm 0.57}$	79.85$_{\pm 0.69}$	84.02$_{\pm 0.61}$	83.47$_{\pm 0.83}$	84.88$_{\pm 0.55}$	84.09$_{\pm 0.86}$	83.82$_{\pm 0.73}$	66.08$_{\pm 22.77}$	85.09$_{\pm 0.71}$	84.41$_{\pm 0.46}$	84.13$_{\pm 0.56}$	**85.09$_{\pm 0.59}$**
ogbn-arxiv	52.80$_{\pm 0.19}$	53.81$_{\pm 0.23}$	70.48$_{\pm 0.18}$	68.77$_{\pm 0.06}$	70.50$_{\pm 0.11}$	69.89$_{\pm 0.59}$	71.04$_{\pm 0.27}$	66.60$_{\pm 0.53}$	**71.48$_{\pm 0.21}$**	68.50$_{\pm 0.17}$	59.28$_{\pm 2.22}$	70.73$_{\pm 0.17}$
Average Rank	9.21	7.86	6.79	7.69	3.50	7.86	7.57	10.50	3.50	4.36	7.29	1.57

Experiment III: Graph Regression

- We utilized the publicly available train/validation/test split of the ZINC-subset dataset.
- For all models, we set the number of hidden units to 64. We limit the training of each model to a maximum of 500 epochs. A batch size of 512 is utilized, and early stopping is implemented after 50 epochs on each dataset.
- We conduct hyperparameter tuning for each model on each dataset and repeat this process 100 times. (Please refer to our paper for the detailed ranges of hyperparameters.)
- To enhance reliability, we repeat the whole process five times and calculate the mean MAE (Mean Absolute Error) and standard deviation for each model.

Table 2. The results of the graph regression experiments.

GCN	GAT	GIN	PointNet	MGNN
0.6143$_{\pm 0.0200}$	0.6458$_{\pm 0.0647}$	**0.4410$_{\pm 0.0065}$**	0.5293$_{\pm 0.0138}$	0.4751$_{\pm 0.0112}$

*Zhewei Wei is the corresponding author. Contact us: cuiguanyu@ruc.edu.cn; zhewei@ruc.edu.cn