BI-LEVEL CONTRASTIVE LEARNING FOR KNOWLEDGE ENHANCED MOLECULE REPRESENTATIONS

Pengcheng Jiang* **Cao Xiao†** **Tianfan Fu*** **Jimeng Sun***
*University of Illinois at Urbana Champaign †GE Healthcare
*{pj20, tianfanf, jimeng}@illinois.edu †danicaxiao@gmail.com

ICLR 2024 under review

Introduction –

- GNNs for representation learning + molecular graph as i/p.
- **[their approach]** combine two types of graph data, molecular + KG based info.
- Previous approaches vs *their method*.
- **G**raph as a N**ode** (**Gode**).

Contributions –

- New Method,
- Robust Embeddings,
- New KG,
- SOTA performance.

Related work –

- GNNs for Molecular Representation Learning,
- Biomedical KGs [PubChemRDF, PrimeKG],
- Molecular Property Prediction **without KGs**,
- Contrastive Learning,
- Fusing KGs and Molecules.

Gode –

- Molecule Graph,

$$G_m = (\mathcal{V}_m, \mathcal{E}_m)$$

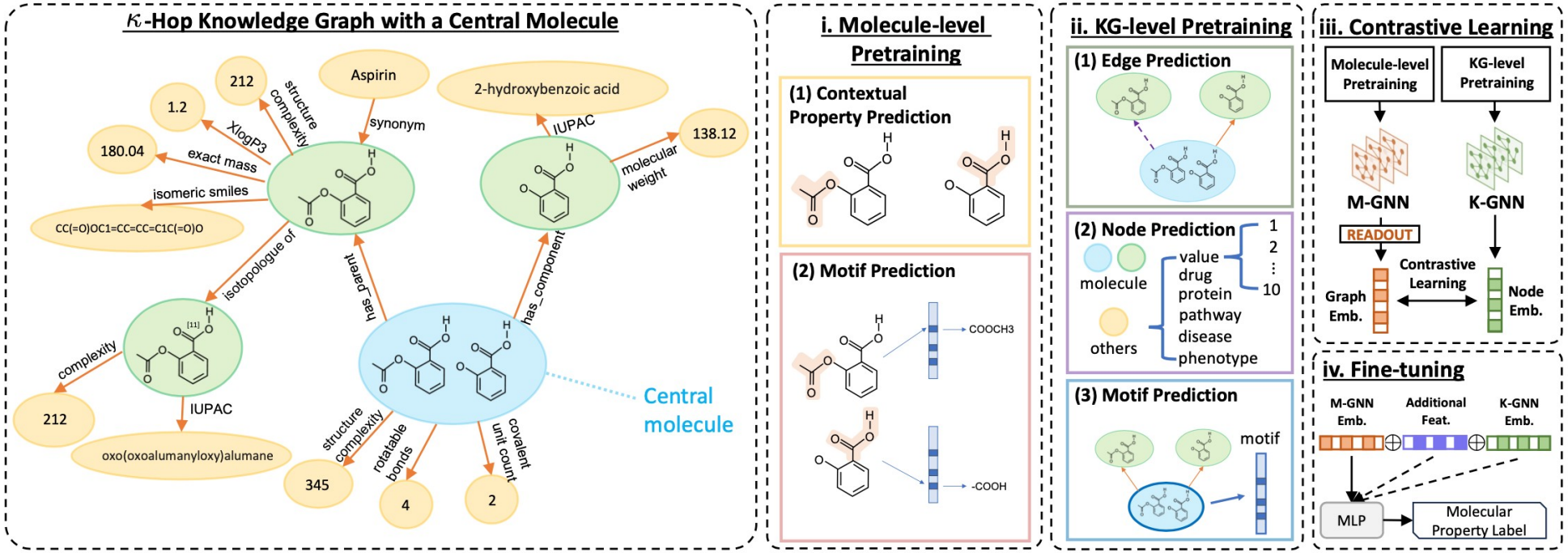
- Knowledge Graph,

$$\mathcal{T} = \{\langle h, r, t \rangle_i\}_i^n$$

Gode –

- M-GNN, $f : \mathcal{M} \rightarrow \mathbb{R}^d$
  which embeds molecules.

- K-GNN, $g : \mathcal{K} \rightarrow \mathbb{R}^d$
  which embeds the central molecule in a KG to an embedding.



Gode –

- *Molecule level Pre-training*

1. Node-level contextual property prediction **[multi-class classification]**,
2. Graph-level motif prediction **[multi-label binary classification]**.

$$\mathcal{L}_{\mathrm{M}} = \sum_{v}^{\mathcal{V}'_m} \log P(p_v|\mathbf{h}_v) + \sum_{j=1}^{n} y_j \log P(M_j|\mathbf{h}_{\mathrm{MG}}) + (1-y_j)\log(1-P(M_j|\mathbf{h}_{\mathrm{MG}})), \quad (1)$$

where $\mathcal{V}'_m$ is a set of randomly selected nodes; p_v is the contextual property label for the node v; n is the number of all possible motifs; M_j is the presence of j-th motif.

Gode –

- *KG level Pre-training*

1. Embedding initialization (e.g. TransE),
2. Sub-Graph Extraction,

$$\mathcal{N}_k(v,h) = \{v\} \cup \bigcup_{u \in \mathcal{N}_k(v,h-1)} \{u\} \cup \bigcup_{u \in \mathcal{M}} \{w : (u,w) \in \mathcal{E}_k\}$$

3. Edge Prediction **[multi-class classification]** predict correct edge-type,
4. Node Prediction **[multi-class classification]** predict type of a node in the sub-graph,
5. Node-level motif-prediction **[multi-label classification]** predict motif of central node.

Gode –

- *KG level Pre-training*

$$\mathcal{L}_{\mathrm{K}} = -\Bigg[\lambda_{\mathrm{edge}} \underbrace{\sum_{(u,v)}^{\mathcal{E}_{\mathrm{sub}(m,\kappa)}} \log P((u,v)'|\mathbf{h}_u \oplus \mathbf{h}_v)}_{\text{edge prediction}} + \lambda_{\mathrm{node}} \underbrace{\sum_{v}^{\mathcal{V}_{\mathrm{sub}(m,\kappa)}} [\log P(v'|\mathbf{h}_v)}_{\text{node prediction}}\Bigg]$$

$$+ \lambda_{\mathrm{mot}} \underbrace{\sum_{j=1}^{n} [y_j \log P(M_j|\mathbf{h}_c) + (1-y_j)\log(1-P(M_j|\mathbf{h}_c))]}_{\text{motif prediction}}, \tag{3}$$

where the first term $(u,v)'$ is the label of edge between the nodes u and v. v' is the label of node v, $\oplus$ denotes the embedding concatenation. y_j is binary indicator, $\log P(M_j|\mathbf{h}_c)$ is the predicted probability of central molecule c has the j-th functional group motif M_j given its embedding $\mathbf{h}_c$. λ_{edge}, λ_{mot}, and λ_{mol} are hyperparameters balancing the importance of different tasks.

Gode –

- *Contrastive learning*

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\text{sim}(f(m_i), g(s_i))) + (1 - y_i)\log(1 - \text{sim}(f(m_i), g(s_i)))\right]$$

$$\text{sim}(f(m_i), g(s_i))) = \frac{\exp\left(\tau^{-1}\mathbf{h}_{\text{MG}(i)}^{\text{T}}\mathbf{h}_{\text{KG}(i)}\right)}{\exp\left(\tau^{-1}\mathbf{h}_{\text{MG}(i)}^{\text{T}}\mathbf{h}_{\text{KG}(i)}\right)+1}$$

- *Fine-tuning for downstream tasks*

$$\mathbf{h}_{\text{joint}} = \mathbf{h}_{\text{MG}} \oplus \mathbf{h}_{\text{f}} \oplus \mathbf{h}_{\text{KG}}$$

Experiments –

Dataset	**BBBP**	**SIDER**	**ClinTox**	**BACE**	**Tox21**	**ToxCast**
# Molecules	2039	1427	1478	1513	7831	8575
# Tasks	1	27	2	1	12	617
GCN (Kipf & Welling, 2016)	71.8 ± 0.9	53.6 ± 0.3	62.5 ± 2.8	71.6 ± 2.0	70.9 ± 0.3	65.0 ± 6.1
GIN (Xu et al., 2018)	65.8 ± 4.5	57.3 ± 1.6	58.0 ± 4.4	70.1 ± 5.4	74.0 ± 0.8	66.7 ± 1.5
Weave (Kearnes et al., 2016)	83.7 ± 6.5	54.3 ± 3.4	82.3 ± 2.3	79.1 ± 0.8	74.1 ± 4.4	67.8 ± 2.4
SchNet (Schütt et al., 2017)	84.8 ± 2.2	54.5 ± 3.8	71.7 ± 4.2	76.6 ± 1.1	76.6 ± 2.5	67.9 ± 2.1
MPNN (Gilmer et al., 2017)	91.3 ± 4.1	59.5 ± 3.0	87.9 ± 5.4	81.5 ± 4.4	80.8 ± 2.4	$\mathbf{69.1 \pm 1.3}$
DMPNN (Yang et al., 2019)	$\mathbf{91.9 \pm 3.0}$	63.2 ± 2.3	89.7 ± 4.0	85.2 ± 5.3	$\mathbf{82.6 \pm 2.3}$	$\mathbf{71.8 \pm 1.1}$
MGCN (Lu et al., 2019)	85.0 ± 6.4	55.2 ± 1.8	63.4 ± 4.2	73.4 ± 3.0	70.7 ± 1.6	66.3 ± 0.9
MGSSL (Zhang et al., 2021)	70.5 ± 1.1	$\mathbf{64.1 \pm 0.7}$	80.7 ± 2.1	79.7 ± 0.8	76.4 ± 0.4	64.1 ± 0.7
N-GRAM (Liu et al., 2019)	91.2 ± 1.3	63.2 ± 0.5	85.5 ± 3.7	$\mathbf{87.6 \pm 3.5}$	76.9 ± 2.7	-
HU. et.al (Hu et al., 2019)	70.8 ± 1.5	62.7 ± 0.8	72.6 ± 1.5	84.5 ± 0.7	78.7 ± 0.4	65.7 ± 0.6
$\text{GROVER}_{\text{Large}}$ (Rong et al., 2020) (our M-GNN)	86.2 ± 3.9	57.6 ± 1.6	74.7 ± 4.4	82.5 ± 4.4	76.9 ± 2.3	66.7 ± 2.6
MolCLR (Wang et al., 2021b)	73.3 ± 1.0	61.2 ± 3.6	$\mathbf{89.8 \pm 2.7}$	82.8 ± 0.7	74.1 ± 5.3	65.9 ± 2.1
KANO (Fang et al., 2023)	$\mathbf{93.7 \pm 2.3}$	$\mathbf{63.8 \pm 1.2}$	$\mathbf{93.6 \pm 0.7}$	$\mathbf{90.4 \pm 1.5}$	$\mathbf{81.2 \pm 1.8}$	$\mathbf{72.5 \pm 1.5}$
GODE (ours)	$\mathbf{94.5 \pm 1.9}$	$\mathbf{67.2 \pm 1.4}$	$\mathbf{94.1 \pm 2.9}$	$\mathbf{91.8 \pm 2.2}$	$\mathbf{84.3 \pm 1.2}$	$\mathbf{73.0 \pm 0.9}$

Experiments –

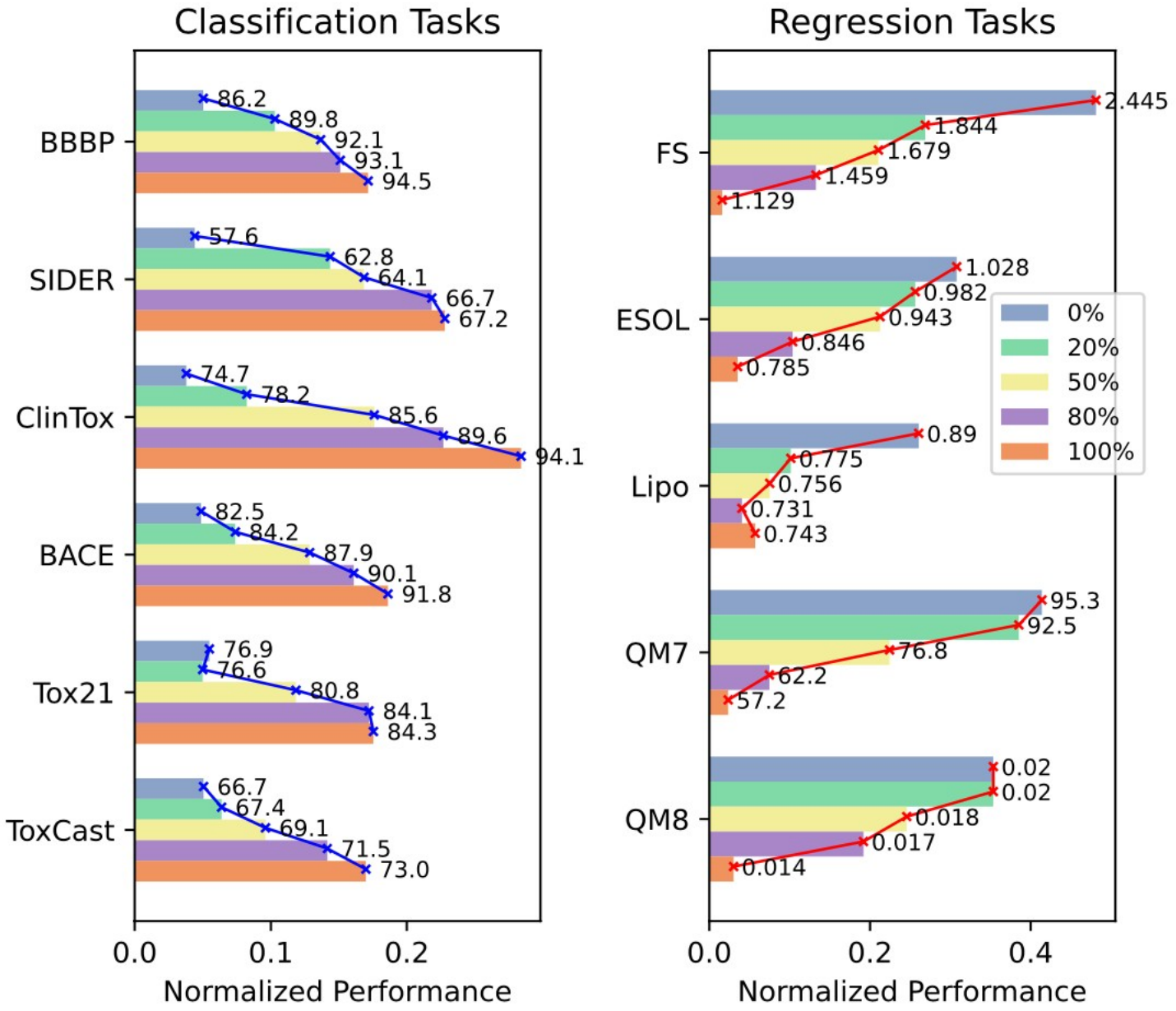
Datasets	**FreeSolv**	**ESOL**	**Lipophilicity**	**QM7**	**QM8**
# Molecules	642	1128	4200	6830	21786
# Tasks	1	1	1	1	12
GCN (Kipf & Welling, 2016)	2.870 ± 0.140	1.430 ± 0.050	0.712 ± 0.049	122.9 ± 2.2	0.037 ± 0.001
GIN (Xu et al., 2018)	2.765 ± 0.180	1.452 ± 0.020	0.850 ± 0.071	124.8 ± 0.7	0.037 ± 0.001
Weave (Kearnes et al., 2016)	2.398 ± 0.250	1.158 ± 0.055	0.813 ± 0.042	94.7 ± 2.7	0.022 ± 0.001
SchNet (Schütt et al., 2017)	3.215 ± 0.755	1.045 ± 0.064	0.909 ± 0.098	$\mathbf{74.2 \pm 6.0}$	0.020 ± 0.002
MPNN (Gilmer et al., 2017)	$\mathbf{1.621 \pm 0.952}$	1.167 ± 0.430	$\mathbf{0.672 \pm 0.051}$	111.4 ± 0.9	$\mathbf{0.015 \pm 0.001}$
DMPNN (Yang et al., 2019)	1.673 ± 0.082	1.050 ± 0.008	$\mathbf{0.683 \pm 0.016}$	103.5 ± 8.6	$\mathbf{0.016 \pm 0.001}$
MGCN (Lu et al., 2019)	3.349 ± 0.097	1.266 ± 0.147	1.113 ± 0.041	77.6 ± 4.7	0.022 ± 0.002
N-GRAM (Liu et al., 2019)	2.512 ± 0.190	1.100 ± 0.160	0.876 ± 0.033	125.6 ± 1.5	0.032 ± 0.003
HU. et.al (Hu et al., 2019)	2.764 ± 0.002	1.100 ± 0.006	0.739 ± 0.003	113.2 ± 0.6	0.022 ± 0.001
GROVER (Rong et al., 2020) (our M-GNN)	2.445 ± 0.761	$\mathbf{1.028 \pm 0.145}$	0.890 ± 0.050	95.3 ± 5.6	0.020 ± 0.003
MolCLR (Wang et al., 2021b)	2.301 ± 0.247	1.113 ± 0.023	0.789 ± 0.009	90.0 ± 1.7	0.019 ± 0.013
KANO (Fang et al., 2023)	$\mathbf{1.443 \pm 0.315}$	$\mathbf{0.914 \pm 0.092}$	$\mathbf{0.651 \pm 0.018}$	$\mathbf{63.6 \pm 4.1}$	$\mathbf{0.014 \pm 0.002}$
GODE (ours)	$\mathbf{1.129 \pm 0.314}$	$\mathbf{0.785 \pm 0.128}$	0.743 ± 0.043	$\mathbf{57.2 \pm 3.0}$	$\mathbf{0.014 \pm 0.001}$

Ablation Study –

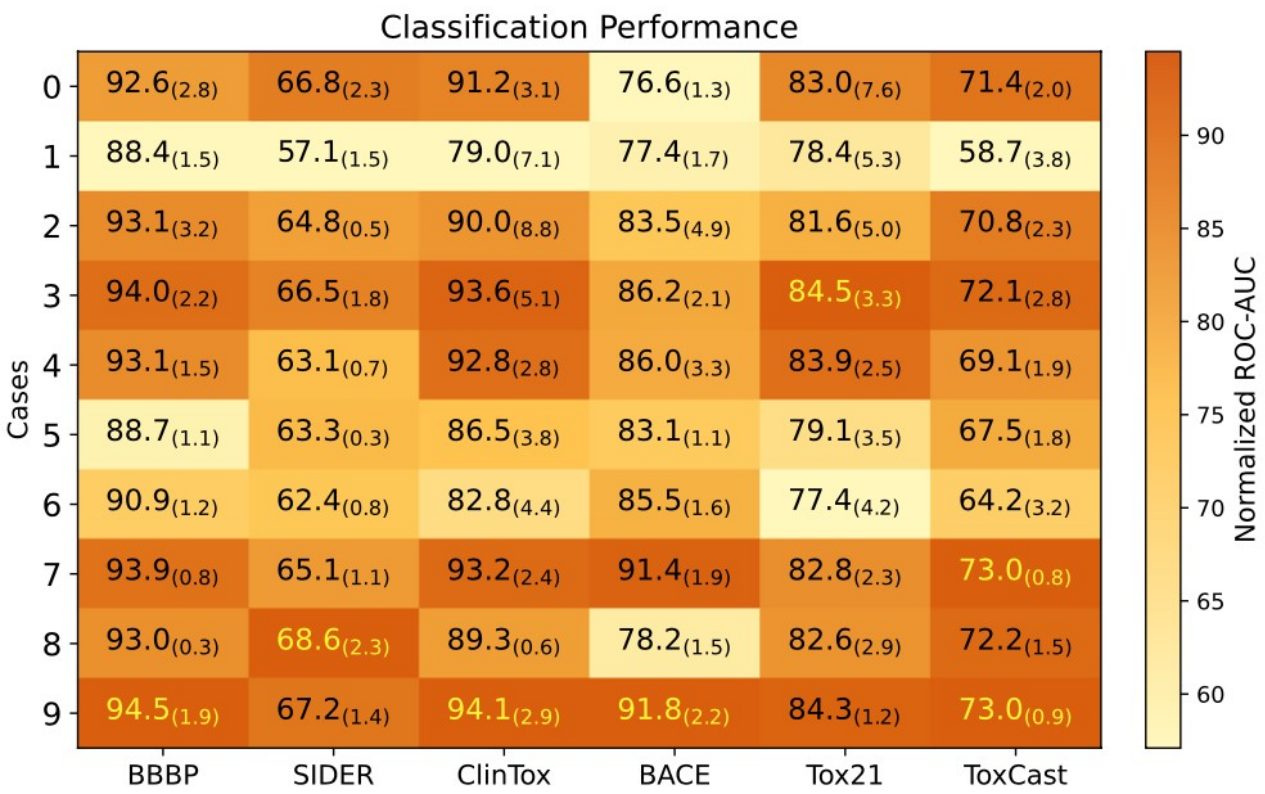
Variants for Ablation Study					
Case	**KGE**	**κ-hop**	**Pret.**	**Cont.**	**Embedding**
⓪	✓	✗	✗	✗	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{KGE}}$
①	✗	$\kappa = 2$	✓	✓	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{KG}}$
②	✓	$\kappa = 2$	✓	✗	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{KG}}$
③	✓	$\kappa = 2$	✓	✓	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{KG}}$
④	✓	$\kappa = 2$	✓	✓	$\mathbf{h}_{\mathrm{MG}}$
⑤	✓	$\kappa = 3$	✓	✓	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{KG}}$
⑥	✓	$\kappa = 3$	✓	✓	$\mathbf{h}_{\mathrm{MG}}$
⑦	✓	$\kappa = 2$	✓	✓	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{f}}$
⑧	✓	✗	✗	✗	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{f}} \oplus \mathbf{h}_{\mathrm{KGE}}$
⑨	✓	$\kappa = 2$	✓	✓	$\mathbf{h}_{\mathrm{MG}} \oplus \mathbf{h}_{\mathrm{f}} \oplus \mathbf{h}_{\mathrm{KG}}$

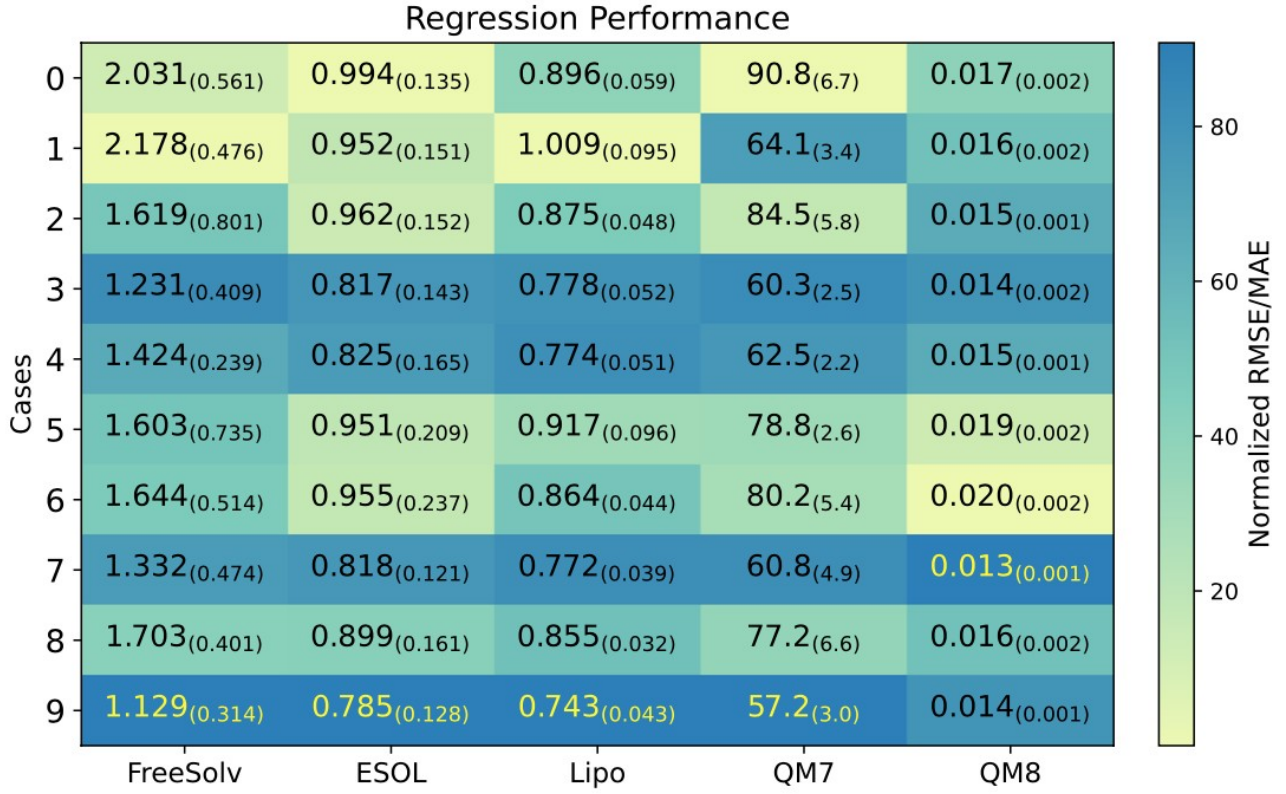
Ablation Study –

- Size of KG.



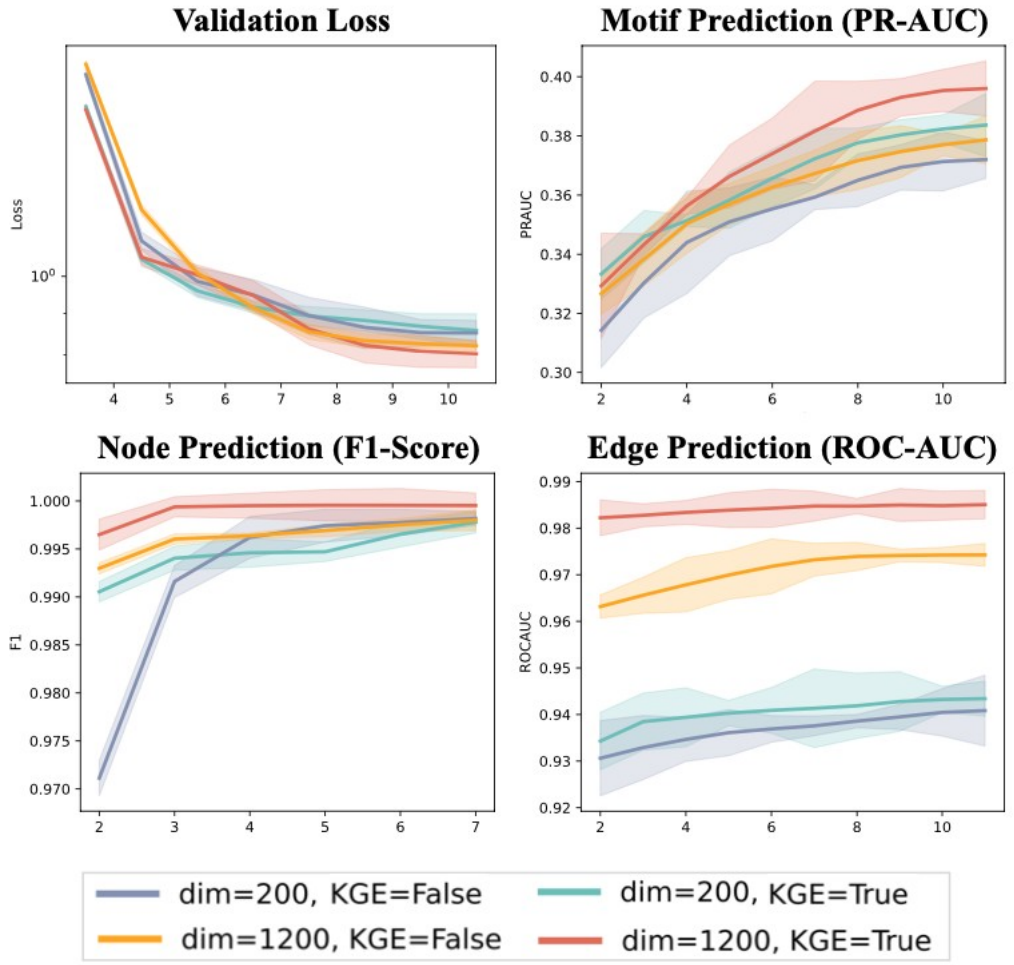
Ablation Study –



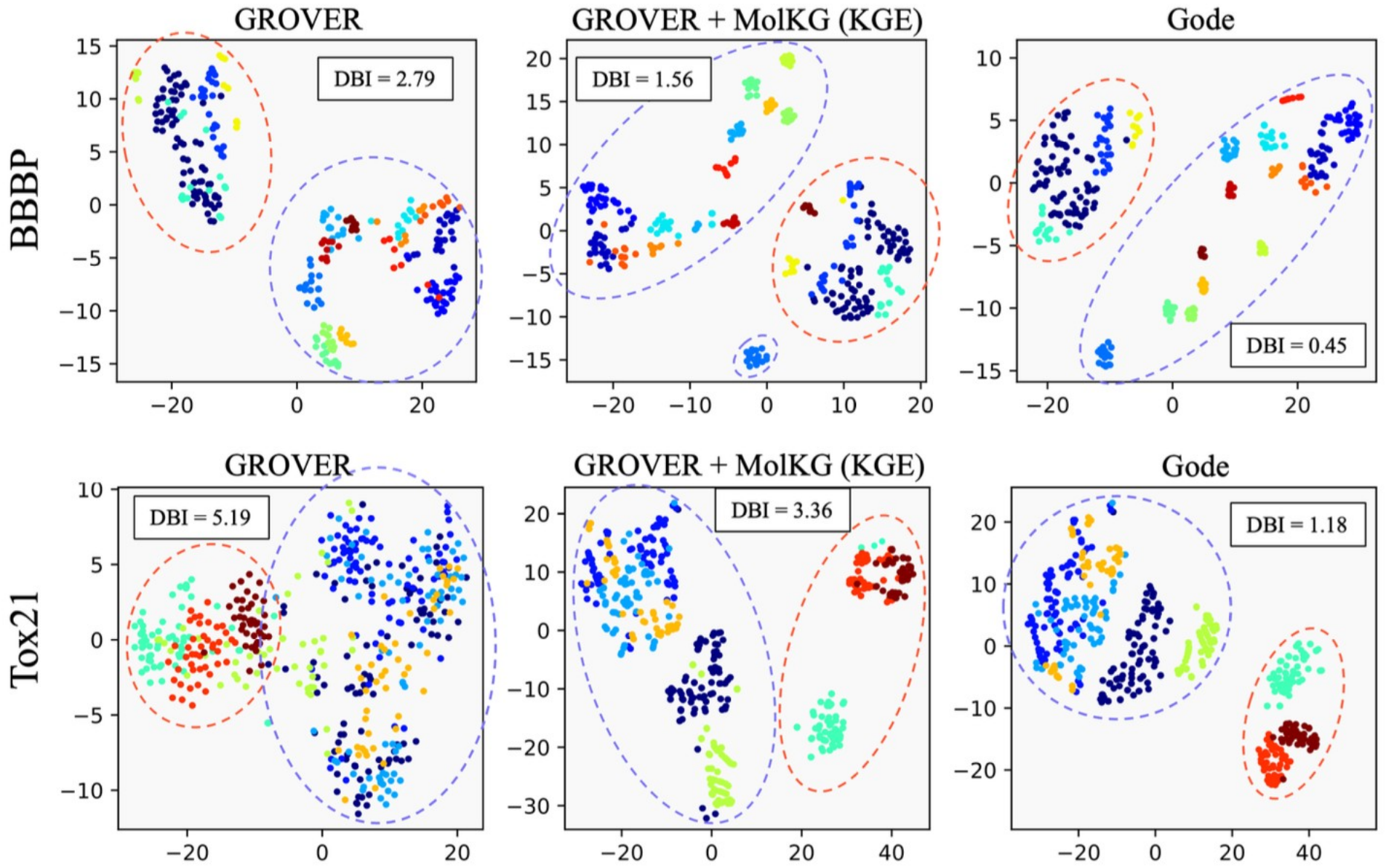


Ablation Study –

- KGE importance.



Ablation Study –



Rank-N-Contrast: Learning Continuous Representations for Regression

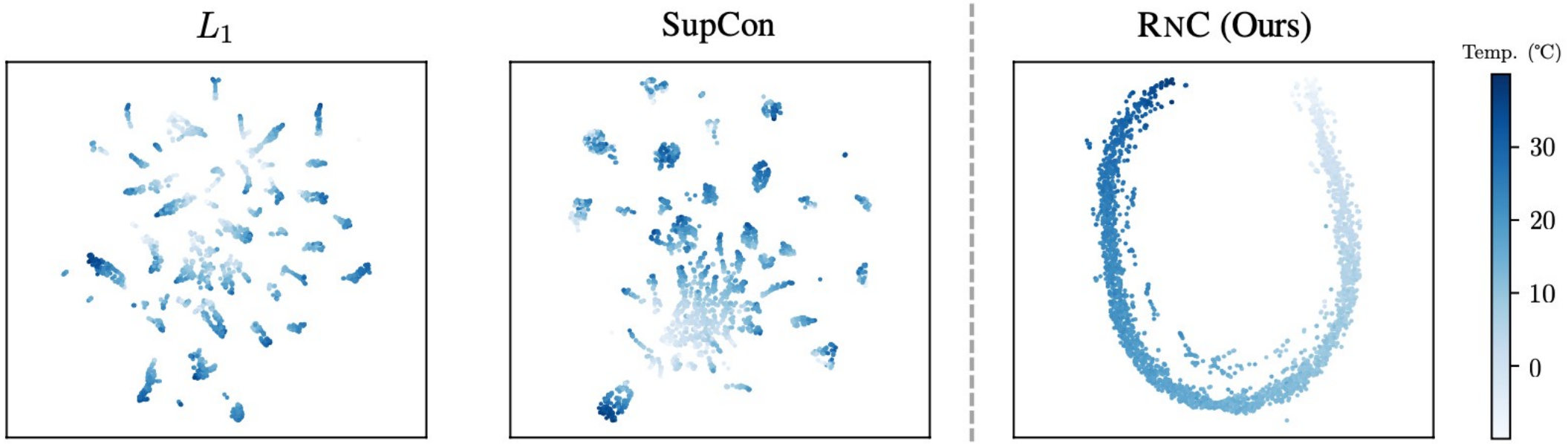
Kaiwen Zha[1,*] **Peng Cao**[1,*] **Jeany Son**[2] **Yuzhe Yang**[1] **Dina Katabi**[1]

[1]MIT CSAIL [2]GIST

NeurIPS 2023 Spotlight

Introduction –

- Current methods for regression tasks, L1/L2.
- *"However, previous methods focus on imposing constraints on the final predictions in an end-to-end fashion, but do not explicitly emphasize the representations learned by the model. Unfortunately, these representations are often fragmented and incapable of capturing the continuous relationships that underlie regression tasks."*



Contributions –

- Identify problems with current methods,
- Propose a new method RnC,
- Extensive experiments that show SOTA performance.

Related work –

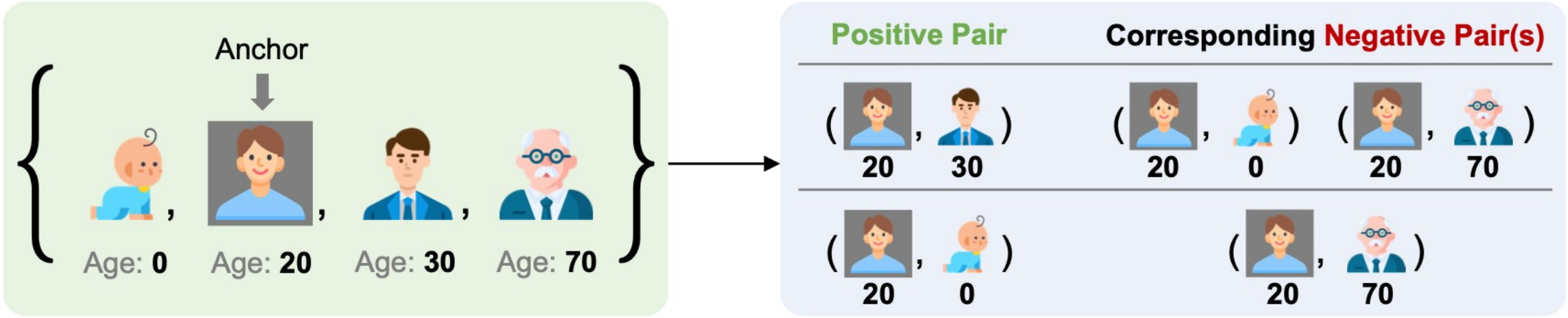
- Normal regression losses like L1, L2, Huber and even binning.
- Representation Learning and SupCon.

Rank-N-Contrast –

1. Augment dataset. Can work without augmenting as well.

$$\tilde{\boldsymbol{x}}_{2n} = t(\boldsymbol{x}_n) \text{ and } \tilde{\boldsymbol{x}}_{2n-1} = t'(\boldsymbol{x}_n)$$

2. Create – and + pairs!



(a) A Batch of Samples

(b) Pair Construction for RNC

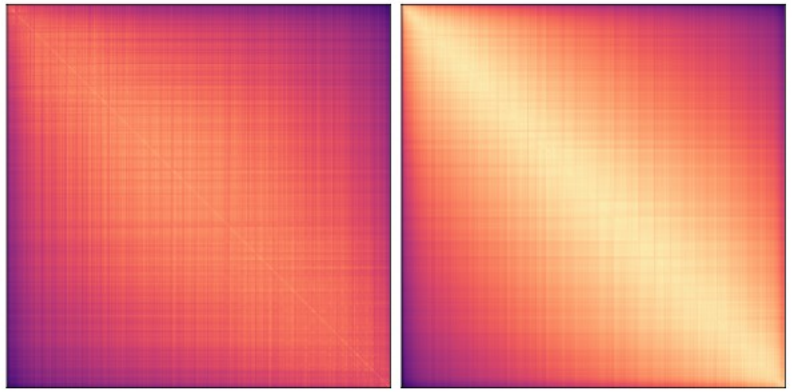
$$\mathcal{S}_{i,j} := \{\boldsymbol{v}_k \mid k \neq i, d(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_k) \geq d(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j)\}$$

Rank-N-Contrast –

3. Loss function, basically, given an anchor, contrast each example with it's negatives to **impose an ordering.**

$$l_{\text{RNC}}^{(i)} = \frac{1}{2N-1} \sum_{j=1,\ j\neq i}^{2N} -\log \frac{\exp(\text{sim}(\boldsymbol{v}_i, \boldsymbol{v}_j)/\tau)}{\sum_{\boldsymbol{v}_k \in \mathcal{S}_{i,j}} \exp(\text{sim}(\boldsymbol{v}_i, \boldsymbol{v}_k)/\tau)}$$

4. Feature Ordinality (data points sorted by ground truth) & Correlation.



L_1 RNC

	Spearman's $\rho^{\uparrow}$	Kendall's $\tau^{\uparrow}$
L_1	0.822	0.664
RNC	**0.971**	**0.870**

Theoretical Analysis –

- They have proved that their loss is **tightly-lower-bounded** by the optimal ordering loss.

Theorem 1 (Lower bound of $\mathcal{L}_{\mathrm{RNC}}$). *$L^\star$ is a lower bound of $\mathcal{L}_{\mathrm{RNC}}$, i.e., $\mathcal{L}_{\mathrm{RNC}} > L^\star$.*

Theorem 2 (Lower bound tightness). *For any $\epsilon > 0$, there exists a set of feature embeddings such that $\mathcal{L}_{\mathrm{RNC}} < L^\star + \epsilon$.*

Theorem 3 (Main theorem). *For any $0 < \delta < 1$, there exist $\epsilon > 0$, such that if $\mathcal{L}_{\mathrm{RNC}} < L^\star + \epsilon$, then the feature embeddings are δ-ordered.*

- Also show that a δ-ordered feature embedding leads to better generalization! So better performance!

Experiments –

- Are based on Vision datasets.

	AgeDB		TUAB		MPIIFaceGaze		SkyFinder	
Metrics	MAE↓	R^{2}↑	MAE↓	R^{2}↑	Angular↓	R^{2}↑	MAE↓	R^{2}↑
L_1	6.63	0.828	7.46	0.655	5.97	0.744	2.95	0.860
RNC(L_1)	**6.14** (+0.49)	**0.850** (+0.022)	**6.97** (+0.49)	**0.697** (+0.042)	**5.27** (+0.70)	**0.815** (+0.071)	2.86 (+0.09)	**0.869** (+0.009)
MSE	6.57	0.828	8.06	0.585	6.02	0.747	3.08	0.851
RNC(MSE)	**6.19** (+0.38)	**0.849** (+0.021)	**7.05** (+1.01)	**0.692** (+0.107)	**5.35** (+0.67)	**0.802** (+0.055)	**2.86** (+0.22)	**0.869** (+0.018)
HUBER	6.54	0.828	7.59	0.637	6.34	0.709	2.92	0.860
RNC(HUBER)	**6.15** (+0.39)	**0.850** (+0.022)	**6.99** (+0.60)	**0.696** (+0.059)	**5.15** (+1.19)	**0.830** (+0.121)	**2.86** (+0.06)	**0.869** (+0.009)
DEX [36]	7.29	0.787	8.01	0.537	5.72	0.776	3.58	0.778
RNC(DEX)	**6.43** (+0.86)	**0.836** (+0.049)	**7.23** (+0.78)	**0.646** (+0.109)	**5.14** (+0.58)	**0.805** (+0.029)	**2.88** (+0.70)	**0.865** (+0.087)
DLDL-v2 [14]	6.60	0.827	7.91	0.560	5.47	0.799	2.99	0.856
RNC(DLDL-v2)	**6.32** (+0.28)	**0.844** (+0.017)	**6.91** (+1.00)	**0.697** (+0.137)	**5.16** (+0.31)	**0.802** (+0.003)	**2.85** (+0.14)	**0.869** (+0.013)
OR [33]	6.40	0.830	7.36	0.646	5.86	0.770	2.92	0.861
RNC(OR)	**6.34** (+0.06)	**0.843** (+0.013)	**7.01** (+0.35)	**0.688** (+0.042)	**5.13** (+0.73)	**0.825** (+0.055)	**2.86** (+0.06)	**0.867** (+0.006)
CORN [40]	6.72	0.811	8.11	0.597	5.88	0.762	3.24	0.819
RNC(CORN)	**6.44** (+0.28)	**0.838** (+0.027)	**7.22** (+0.89)	**0.663** (+0.066)	**5.18** (+0.70)	**0.820** (+0.058)	**2.89** (+0.35)	**0.862** (+0.043)

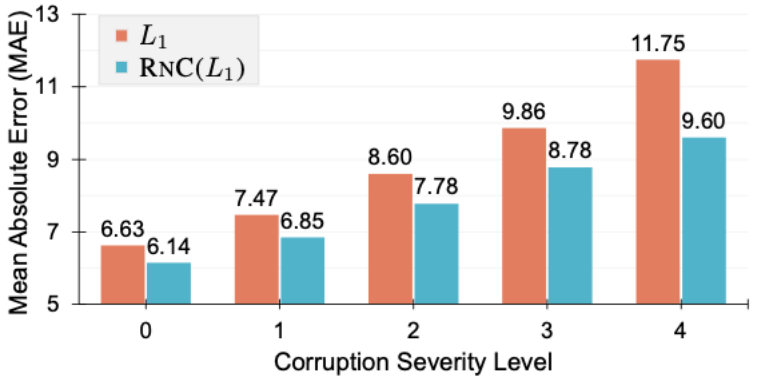
Experiments –

- Comparison to SOTA.

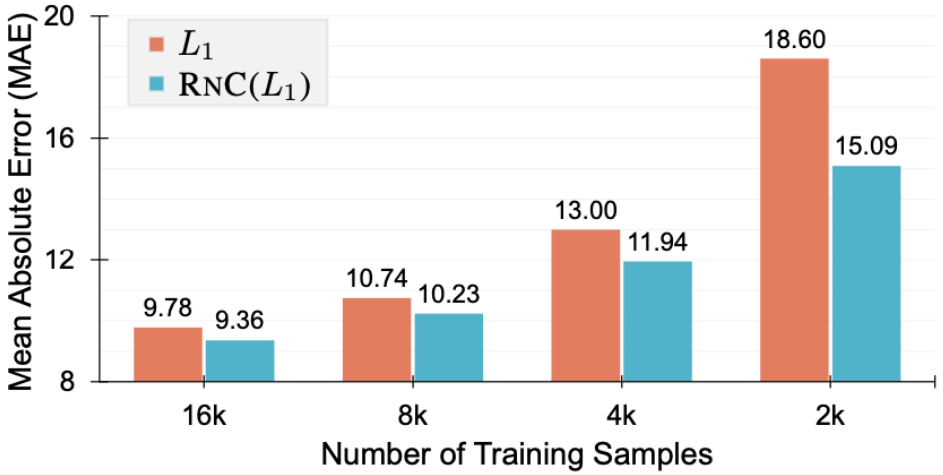
Method	AgeDB	TUAB	MPIIFaceGaze	SkyFinder
Representation learning methods (Linear Probing):				
SimCLR [4]	9.59	11.01	9.43	4.70
DINO [3]	10.26	11.62	11.92	5.63
SupCon [25]	8.13	8.47	9.27	3.97
Representation learning methods (Fine-tuning):				
SimCLR [4]	6.57	7.57	5.50	2.93
DINO [3]	6.61	7.58	5.80	2.98
SupCon [25]	6.55	7.41	5.54	2.95
Regression learning methods:				
L_1	6.63	7.46	5.97	2.95
LDS+FDS [44]	6.45	–	–	–
L2CS-Net [1]	–	–	5.45	–
LDE [7]	–	–	–	2.92
RankSim [17]	6.51	7.33	5.70	2.94
Ordinal Entropy [50]	6.47	7.28	–	2.94
RnC(L_1)	**6.14**	**6.97**	**5.27**	**2.86**
Gains	**+0.31**	**+0.31**	**+0.18**	**+0.06**

Experiments –

- Robustness to data corruption.



- Resilience to reduced dataset size.

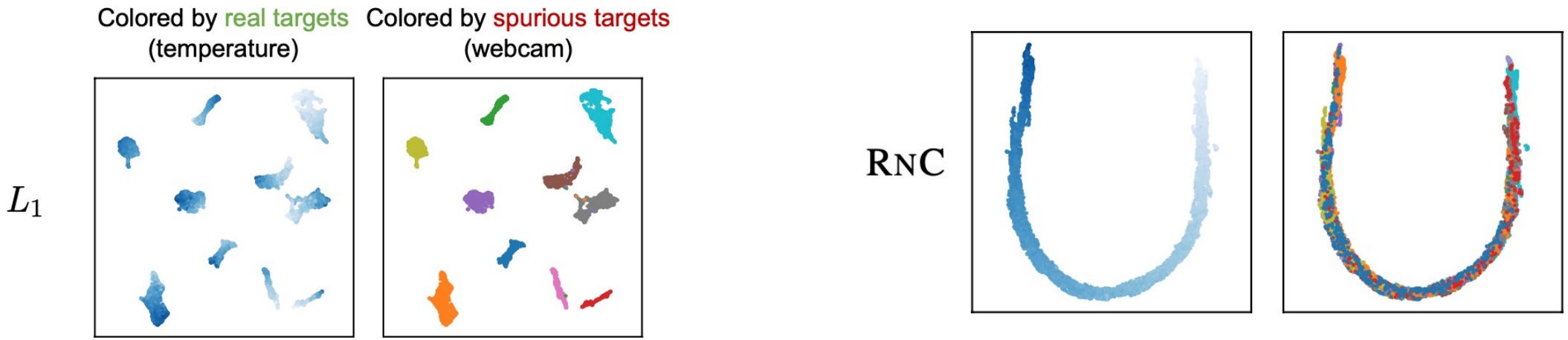


Experiments –

- Transfer Learning.

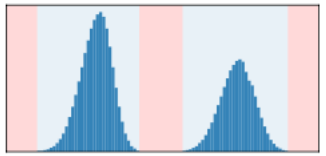
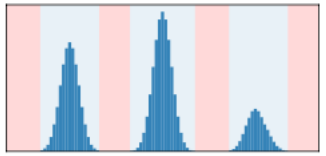
	AgeDB → IMDB-WIKI (subsampled, 2k)				IMDB-WIKI (subsampled, 32k) → AgeDB			
	Linear Probing		*Fine-tuning*		*Linear Probing*		*Fine-tuning*	
Metrics	MAE↓	R^{2}↑	MAE↓	R^{2}↑	MAE↓	R^{2}↑	MAE↓	R^{2}↑
L_1	12.25	0.496	11.57	0.528	7.36	0.801	6.36	0.848
RNC(L_1)	**11.12** (+1.13)	**0.556** (+0.060)	**11.09** (+0.48)	**0.546** (+0.018)	**7.06** (+0.30)	**0.812** (+0.011)	**6.13** (+0.23)	**0.850** (+0.002)

- Robustness to spurious targets.



Experiments –

- Zero-shot generalization.

Label Distribution	Method	All	Seen	Unseen
	L_1	12.53	10.82	18.40
	RNC(L_1)	**11.69**	**10.46**	**15.92**
		(+0.84)	**(+0.36)**	**(+2.48)**
	L_1	11.94	10.43	14.98
	RNC(L_1)	**10.88**	**9.78**	**13.08**
		(+1.06)	**(+0.64)**	**(+1.90)**

Ablation Study –

- Number of positives.

(a) Number of Positives K

	MAE$^{\downarrow}$	$R^{2\uparrow}$
128	6.46	0.828
256	6.43	0.833
384	6.29	0.845
511	**6.14**	**0.850**

Ablation Study –

- Sim(., .) function effect.

	MAE$^{\downarrow}$	$R^{2\uparrow}$
cosine	6.51	0.836
negative L_1 norm	6.25	0.842
negative L_2 norm	**6.14**	**0.850**

- Effect of training scheme.

	MAE$^{\downarrow}$	$R^{2\uparrow}$
linear probing	**6.14**	**0.850**
fine-tuning	6.36	0.844
regularization	6.42	0.833

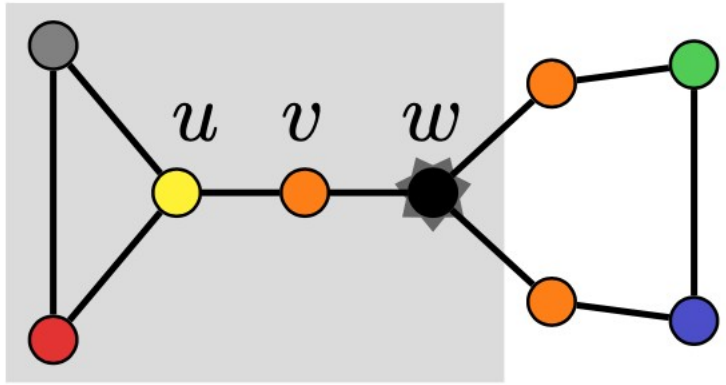
COOPERATIVE GRAPH NEURAL NETWORKS

Ben Finkelshtein, Xingyue Huang, Michael Bronstein, İsmail İlkan Ceylan
Department of Computer Science
University of Oxford
{name.surname}@cs.ox.ac.uk

(maybe) ICLR 2024 under review

Introduction –

- Problems with usual GNNs,
- Generalized MPNNs,



- Each node as a **player** with 4 possible actions,

STANDARD (S): Broadcast to neighbors that listen *and* listen to neighbors that broadcast.

LISTEN (L): Listen to neighbors that broadcast.

BROADCAST (B): Broadcast to neighbors that listen.

ISOLATE (I): Neither listen nor broadcast, effectively isolating the node.

Contributions –

- Novel problem formulation with an **environment network** η and an **action network** π which allows a **learnable topology**.
- Given approach is **more expressive** than the 1D WL algorithm and better-suited for longer-range tasks.
- SOTA performance.

Background –

- Graph Neural Networks,
  generalized – $\boldsymbol{h}_v^{(\ell+1)} = \phi^{(\ell)}\left(\boldsymbol{h}_v^{(\ell)}, \psi^{(\ell)}\left(\boldsymbol{h}_v^{(\ell)}, \{\!\{\boldsymbol{h}_u^{(\ell)} \mid u \in \mathcal{N}_v\}\!\}\right)\right)$
  their focus – $\boldsymbol{h}_v^{(\ell+1)} = \sigma\left(\boldsymbol{W}_s^{(\ell)}\boldsymbol{h}_v^{(\ell)} + \boldsymbol{W}_n^{(\ell)}\psi\left(\{\!\{\boldsymbol{h}_u^{(\ell)} \mid u \in \mathcal{N}_v\}\!\}\right)\right)$
  and prominent ones like GIN, GCN, etc.

Background –

- Straight-through Gumbel Softmax!

$$\text{Gumbel-softmax}\left(\boldsymbol{p};\tau\right) = \frac{\exp\left(\left(\log(\boldsymbol{p}) + \boldsymbol{g}\right)/\tau\right)}{\sum_{a\in\Omega}\exp\left(\left(\log(\boldsymbol{p}(a)) + \boldsymbol{g}(a)\right)/\tau\right)}$$

Related Work –

- Standard GNNs and problems (e.g. limited by 1-WL test),
- Recent transformer-GNN surge,
- Synchronous message passing,
- Orthogonal work on picking optimal depth for each node using RL.

Co-GNNs –

- Basic idea – every node as a **player in an multiplayer environment**,
- 2-stage update process,

first – $\boldsymbol{p}_v^{(\ell)} = \pi\left(\boldsymbol{h}_v^{(\ell)}, \{\!\{\boldsymbol{h}_u^{(\ell)} \mid u \in \mathcal{N}_v\}\!\}\right)$

second – $\boldsymbol{h}_v^{(\ell+1)} = \begin{cases} \eta^{(\ell)}\left(\boldsymbol{h}_v^{(\ell)}, \{\!\{\}\!\}\right), & a_v^{(\ell)} = \mathrm{I} \vee \mathrm{B} \\ \eta^{(\ell)}\left(\boldsymbol{h}_v^{(\ell)}, \{\!\{\boldsymbol{h}_u^{(\ell)} \mid u \in \mathcal{N}_v, a_u^{(\ell)} = \mathrm{S} \vee \mathrm{B}\}\!\}\right), & a_v^{(\ell)} = \mathrm{L} \vee \mathrm{S} \end{cases}$

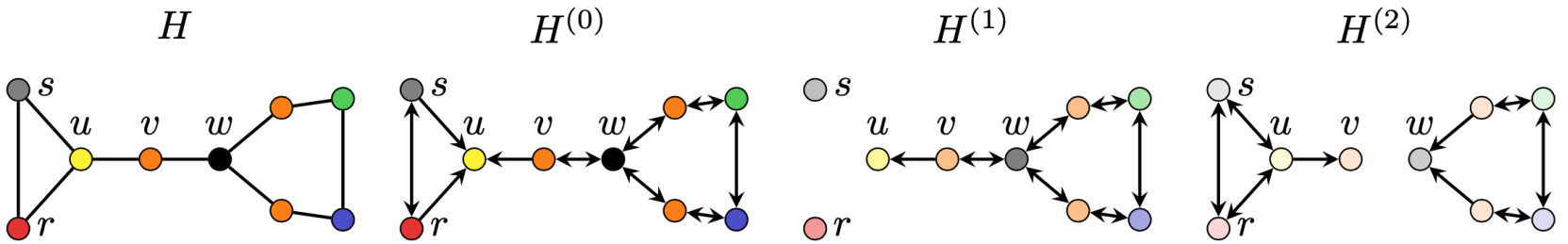
- If Isolate (I) or Broadcast (B) is chosen, a node is updated only using it's state;
  else (when Listen (L) or Standard(S)) is chosen, a node is updated using a normal GNN.
- π and η can be **any** GNN!

Model Properties –

1. Task-specific learnable computation graph.
2. Can re-structure edges on different levels, leading to different graphs at each level!

$$\boldsymbol{h}_v^{(\ell+1)} = \eta^{(\ell)}\left(\boldsymbol{h}_v^{(\ell)}, \{\!\{\boldsymbol{h}_u^{(\ell)} \mid (u,v) \in E^{(\ell)}\}\!\}\right)$$

3. Dynamic computation graph across layers.
4. Feature + Structure based!
5. Asynchronous.
6. Efficient. O(GCN)!



(a) Input graph H. (b) Computation at $\ell = 0$. (c) Computation at $\ell = 1$. (d) Computation at $\ell = 2$.

Figure 3: The input graph H and its computation graphs $H^{(0)}$, $H^{(1)}$, $H^{(2)}$ at the respective layers. The computation graphs are a result of applying the following actions: $\langle \mathrm{L}, \mathrm{L}, \mathrm{S} \rangle$ for the node u; $\langle \mathrm{S}, \mathrm{S}, \mathrm{L} \rangle$ for the nodes v and w; $\langle \mathrm{S}, \mathrm{I}, \mathrm{S} \rangle$ for the nodes s and r; $\langle \mathrm{S}, \mathrm{S}, \mathrm{S} \rangle$ for all other nodes.

Propositions –

- > 1-WL test.

Proposition 5.1. *Let* $G_1 = (V_1, E_1, \boldsymbol{X}_1)$ *and* $G_2 = (V_2, E_2, \boldsymbol{X}_2)$ *be two non-isomorphic graphs. Then, for any threshold* $0 < \delta < 1$*, there exists a parametrization of a* Co-GNN *architecture using sufficiently many layers* L*, satisfying* $\mathbb{P}(\boldsymbol{z}_{G_1}^{(L)} \neq \boldsymbol{z}_{G_2}^{(L)}) \geq 1 - \delta$.

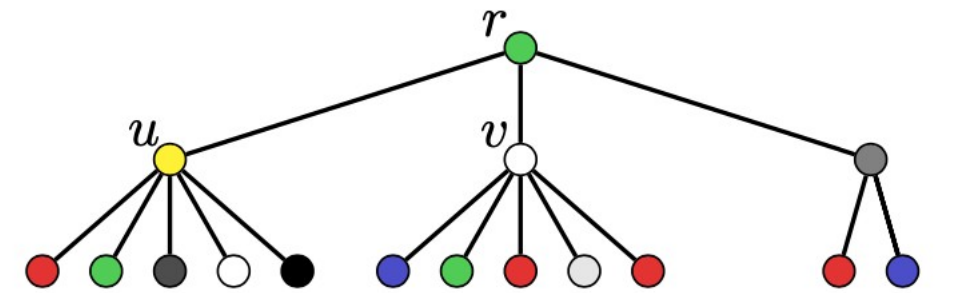
- Dynamic long-range message passing.

Theorem 5.2. *Let* $G = (V, E, \boldsymbol{X})$ *be a connected graph with node features. For some* $k > 0$*, for any target node* $v \in V$*, for any* k *source nodes* $u_1, \ldots, u_k \in V$*, and for any compact, differentiable function* $f : \mathbb{R}^{d^{(0)}} \times \ldots \times \mathbb{R}^{d^{(0)}} \to \mathbb{R}^d$*, there exists an* L*-layer* Co-GNN *computing final node representations such that for any* $\epsilon, \delta > 0$ *it holds that* $\mathbb{P}(|\boldsymbol{h}_v^{(L)} - f(\boldsymbol{x}_{u_1}, \ldots \boldsymbol{x}_{u_k})| < \epsilon) \geq 1 - \delta$.

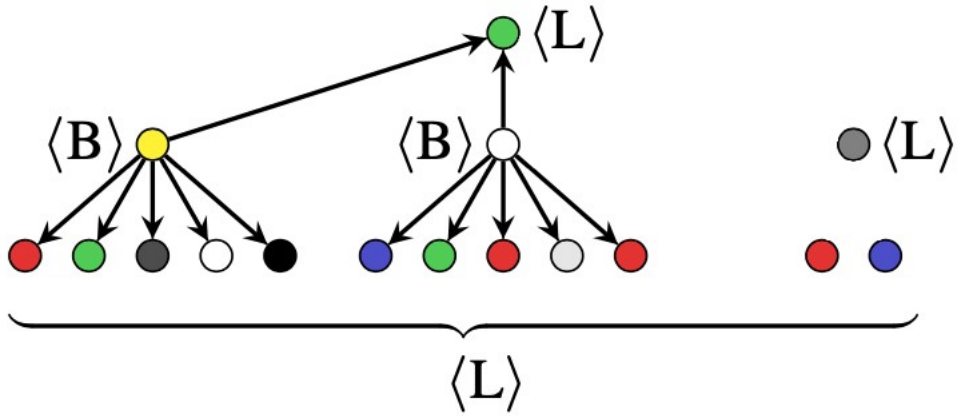
- Prevents both **over-squashing** and **over-smoothening**!

Experiments –

- Synthetic experiment, new task RootNeighbors, *"given a rooted tree, predict the average of the features of root-neighbors of degree 6".*



(a) Example tree for ROOTNEIGHBORS.



Experiments –

Model	MAE
Random	0.474
GCN	0.468
SAGE	0.336
GAT	0.442
SUMGNN	0.370
MEANGNN	0.329
CO-GNN(Σ, Σ)	0.196
CO-GNN(μ, μ)	0.339
CO-GNN(Σ, μ)	0.079

1. The action network chooses either the action LISTEN or STANDARD for the root node, and the action BROADCAST or STANDARD for the root-neighbors which have a degree 6,
2. The action network chooses either the action LISTEN or the action ISOLATE for all the remaining root-neighbors, and
3. The environment network updates the root node by averaging the features of its neighbors which are currently broadcasting.

- Why is MeanGNN better? Applies a different non-linearity on the source node!

Experiments –

- Node Classification (all are heterophilic datasets).

	roman-empire	amazon-ratings	minesweeper	tolokers	questions
GCN	73.69 ± 0.74	48.70 ± 0.63	89.75 ± 0.52	83.64 ± 0.67	76.09 ± 1.27
SAGE	85.74 ± 0.67	53.63 ± 0.39	93.51 ± 0.57	82.43 ± 0.44	76.44 ± 0.62
GAT	80.87 ± 0.30	49.09 ± 0.63	92.01 ± 0.68	83.70 ± 0.47	77.43 ± 1.20
GAT-sep	88.75 ± 0.41	52.70 ± 0.62	93.91 ± 0.35	83.78 ± 0.43	76.79 ± 0.71
GT	86.51 ± 0.73	51.17 ± 0.66	91.85 ± 0.76	83.23 ± 0.64	77.95 ± 0.68
GT-sep	87.32 ± 0.39	52.18 ± 0.80	92.29 ± 0.47	82.52 ± 0.92	78.05 ± 0.93
Co-GNN(Σ, Σ)	91.57 ± 0.32	51.28 ± 0.56	95.09 ± 1.18	83.36 ± 0.89	80.02 ± 0.86
Co-GNN(μ, μ)	91.37 ± 0.35	54.17 ± 0.37	97.31 ± 0.41	84.45 ± 1.17	76.54 ± 0.95

Experiments –

- Graph Classification.

	IMDB-B	IMDB-M	REDDIT-B	NCI1	PROTEINS	ENZYMES
DGCNN	69.2 ± 3.0	45.6 ± 3.4	87.8 ± 2.5	76.4 ± 1.7	72.9 ± 3.5	38.9 ± 5.7
DiffPool	68.4 ± 3.3	45.6 ± 3.4	89.1 ± 1.6	76.9 ± 1.9	73.7 ± 3.5	59.5 ± 5.6
ECC	67.7 ± 2.8	43.5 ± 3.1	OOR	76.2 ± 1.4	72.3 ± 3.4	29.5 ± 8.2
GIN	71.2 ± 3.9	48.5 ± 3.3	89.9 ± 1.9	80.0 ± 1.4	73.3 ± 4.0	59.6 ± 4.5
GraphSAGE	68.8 ± 4.5	47.6 ± 3.5	84.3 ± 1.9	76.0 ± 1.8	73.0 ± 4.5	58.2 ± 6.0
ICGMM_f	71.8 ± 4.4	49.0 ± 3.8	91.6 ± 2.1	76.4 ± 1.4	73.2 ± 3.9	-
SPN($k = 5$)	-	-	-	78.6 ± 1.7	74.2 ± 2.7	69.4 ± 6.2
Co-GNN(Σ, Σ)	70.8 ± 3.3	48.5 ± 4.0	88.6 ± 2.2	80.6 ± 1.1	73.1 ± 2.3	65.7 ± 4.9
Co-GNN(μ, μ)	72.2 ± 4.1	49.9 ± 4.5	90.4 ± 1.9	79.4 ± 0.7	71.3 ± 2.0	68.3 ± 5.7

Idea –

1. One layer of Molecular aggregation, one layer of KG stuff.
2. Molecular aggregation => Spec/Graphormer(Mol. Graph + [VNode])
3. KG stuff => use [VNode] as mol embedding and use HyNT.
4. Pre train using final embedding as in Gode.

5. Possible alterations – all layers of Mol then all layers of KG.
6. Make [VNode] big and share across all molecules.

Novelty –

- Joint optimization and free interaction of both modalities like this has not been explored.

Problems –

1. No GNNs?
2. Can use Co-GNNs in some way to allow which edges should be present?
3. Are molecule graphs heterogeneous? Then cannot use Specformer as is.