On oversquashing and expressivity: can GNNs mix variables? From theory to physics-inspired solutions

Francesco Di Giovanni

University of Cambridge

Complex Network Analysis Discussion Group 12 June 2023

Overview

- *How does over-squashing affect the power of GNNs?* Novel approach to expressive power relating it to over-squashing
  F. Di G.*, T. K. Rusch*, M. Bronstein, A. Deac, M. Lackenby, S. Mishra, P. Veličković (2023)

- *DRew: dynamically rewired message passing with delay* Novel framework for message-passing that changes where and when messages are exchanged
  B. Gutteridge, X. Dong, M. Bronstein, **F. Di G.**, ICML2023

Notations and conventions: a useful shopping list

- Graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n nodes and $\mathbf{A} \in \mathbb{R}^{n \times n}$ s.t. $(v, u) \in \mathsf{E} \iff A_{vu} > 0$
- The diagonal degree matrix is $\mathbf{D} = \mathrm{diag}(d_v)$, with $d_v = \sum_u A_{vu}$.
- Node-wise features $X : \mathsf{V} \to \mathbb{R}^p$ with matrix representation $\mathbf{X} \in \mathbb{R}^{n \times p}$
- $\mathbf{h}_v^{(t)}$ represents the (hidden) feature of node v at layer t

How does over-squashing affect the power of GNNs?

The class of MPNNs

We consider Message Passing Neural Networks (MPNNs) of the form:

$$\mathbf{h}_v^{(t)} = \sigma\Big(\mathbf{\Omega}^{(t)}\mathbf{h}_v^{(t-1)} + \mathbf{W}^{(t)}\sum_u \mathsf{A}_{vu}\psi^{(t)}(\mathbf{h}_v^{(t-1)}, \mathbf{h}_u^{(t-1)})\Big), \quad \mathbf{h}_v^{(0)} = \mathbf{x}_v$$

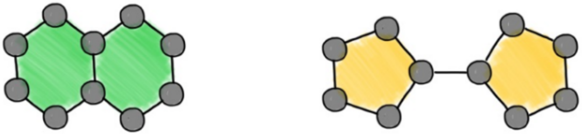
- σ is a pointwise nonlinear map, $\mathbf{\Omega}^{(t)}$, $\mathbf{W}^{(t)}$ are learnable weight matrices
- A is a message-passing matrix typically chosen in $\{\mathbf{D}^{-1}\mathbf{A}, \mathbf{A}, \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\}$
- $\psi^{(t)}$ is a learnable message-passing function

$\rightsquigarrow$ include standard models as GCN, GraphSAGE, GIN, GatedGCN

Assumption: All non-linear functions involved are in $\mathcal{C}^2$

Expressive power

- Expressive power[1] $\to$ typically studied through the lenses of WL-algorithms

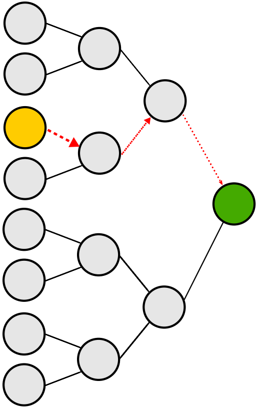


- What about **features**?

Which classes of functions can MPNNs of a given capacity learn, if node features are specified?

[1] Xu et al. (2019)

Over-squashing



- *Depending on the graph-topology but independent of the chosen* MPNN, the size of the $r-$hop of a node may grow **exponentially**

- As messages are sent through fixed-dimension channels, node green is no longer *sensitive* to information contained at node yellow

How does over-squashing affect the expressive power of MPNNs? Can we measure it?

Readout and graph-level predictions

Consider a graph-regression task: $y_{\mathsf{G}} : \mathbb{R}^{n \times p} \to \mathbb{R}$

Given an MPNN of m layers $\to$ generate prediction $y_{\mathsf{G}}^{(m)} = \theta^\top \mathsf{READ}(\{\!\{\mathbf{h}_v^{(m)}\}\!\})\rangle$

Definition

The maximal mixing induced by an MPNN of m layers among the features $\mathbf{x}_v$ and $\mathbf{x}_u$ is

$$\mathsf{mix}_{y_{\mathsf{G}}^{(m)}}(v,u) = \max_{\mathbf{X}} \max_{1 \leq \alpha, \beta \leq d} \left| \frac{\partial^2 y_{\mathsf{G}}^{(m)}(\mathbf{X})}{\partial x_v^\alpha \partial x_u^\beta} \right| .$$

- If $y_{\mathsf{G}}(\mathbf{x}_v, \mathbf{x}_u) = \phi_0(\mathbf{x}_v) + \phi_1(\mathbf{x}_u)$, then low-mixing MPNNs can solve the task
- If $y_{\mathsf{G}}(\mathbf{x}_v, \mathbf{x}_u) = \phi(\mathbf{x}_u^\top \mathbf{x}_v)$, with ϕ highly nonlinear, only high-mixing MPNNs can solve the task

Some intuition

Take a *linear* MPNN with $p = 1$, weights $\omega, \mathsf{w} > 0$ and $\psi(x, y) = c_1 x + c_2 y \implies$

$$h_v^{(t)} = (\mathsf{w}\mathbf{S}\mathbf{h}^{(t-1)})_v, \quad \mathbf{S} := \frac{\omega}{\mathsf{w}}\mathbf{I} + c_1 \mathrm{diag}(\mathbf{A}\mathbf{1}) + c_2 \mathbf{A} \in \mathbb{R}^{n \times n},$$

- $(\mathsf{w}\mathbf{S})^k$ determines info flow over k layers, i.e. over walks of length k

Some intuition

Take a *linear* MPNN with $p = 1$, weights $\omega, \mathsf{w} > 0$ and $\psi(x, y) = c_1 x + c_2 y \implies$

$$h_v^{(t)} = (\mathsf{w}\mathbf{S}\mathbf{h}^{(t-1)})_v, \quad \mathbf{S} := \frac{\omega}{\mathsf{w}}\mathbf{I} + c_1 \mathrm{diag}(\mathbf{A}\mathbf{1}) + c_2 \mathbf{A} \in \mathbb{R}^{n \times n},$$

- $(\mathsf{w}\mathbf{S})^k$ determines info flow over k layers, i.e. over walks of length k
- Set $d_{\mathsf{G}}(v, u) = r$; if $m < r/2$, then $\mathrm{mix}_{y_{\mathsf{G}}^{(m)}(v,u)}(v, u) = 0$

Some intuition

Take a *linear* MPNN with $p = 1$, weights $\omega, \mathsf{w} > 0$ and $\psi(x, y) = c_1 x + c_2 y \implies$

$$h_v^{(t)} = (\mathsf{w}\mathbf{S}\mathbf{h}^{(t-1)})_v, \quad \mathbf{S} := \frac{\omega}{\mathsf{w}}\mathbf{I} + c_1 \mathrm{diag}(\mathbf{A}\mathbf{1}) + c_2 \mathbf{A} \in \mathbb{R}^{n \times n},$$

- $(\mathsf{w}\mathbf{S})^k$ determines info flow over k layers, i.e. over walks of length k
- Set $d_{\mathsf{G}}(v, u) = r$; if $m < r/2$, then $\mathsf{mix}_{y_{\mathsf{G}}^{(m)}(v,u)}(v, u) = 0$
- If $m = r/2$, then we can estimate the mixing at i s.t. $d_{\mathsf{G}}(v, i) = d_{\mathsf{G}}(u, i)$ as $((\mathsf{w}\mathbf{S})^m)_{iv}((\mathsf{w}\mathbf{S})^m)_{iu}$.

Some intuition

Take a *linear* MPNN with $p = 1$, weights $\omega, \mathsf{w} > 0$ and $\psi(x, y) = c_1 x + c_2 y \implies$

$$h_v^{(t)} = (\mathsf{w}\mathbf{S}\mathbf{h}^{(t-1)})_v, \quad \mathbf{S} := \frac{\omega}{\mathsf{w}}\mathbf{I} + c_1 \mathrm{diag}(\mathbf{A}\mathbf{1}) + c_2 \mathbf{A} \in \mathbb{R}^{n \times n},$$

- $(\mathsf{w}\mathbf{S})^k$ determines info flow over k layers, i.e. over walks of length k
- Set $d_{\mathsf{G}}(v, u) = r$; if $m < r/2$, then $\mathsf{mix}_{y_{\mathsf{G}}^{(m)}(v,u)}(v, u) = 0$
- If $m = r/2$, then we can estimate the mixing at i s.t. $d_{\mathsf{G}}(v, i) = d_{\mathsf{G}}(u, i)$ as $((\mathsf{w}\mathbf{S})^m)_{iv}((\mathsf{w}\mathbf{S})^m)_{iu}$.
- The general case: $|\sigma'|, |\sigma''| \leq c_\sigma$, $\|\mathbf{\Omega}^{(t)}\| \leq \omega$, $\|\mathbf{W}^{(t)}\| \leq \mathsf{w}$, $\|\nabla_i \psi^{(t)}\| \leq c_i$, $\|\nabla^2 \psi^{(t)}\| \leq c^{(2)}$

The maximal mixing induced by MPNNs

Theorem

Consider an MPNN of depth m with bounds on the derivatives and on the norm of the weights as above. If the readout is MAX, MEAN *or* SUM *then the mixing* $\mathsf{mix}_{y_{\mathsf{G}}^{(m)}}(v,u)$ *induced by the MPNN over $\mathbf{x}_v, \mathbf{x}_u$ satisfies*

$$\mathsf{mix}_{y_{\mathsf{G}}^{(m)}}(v,u) \leq \sum_{k=0}^{m-1} (c_\sigma \mathsf{w})^{2m-k-1} \left(\mathsf{w} (\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k) \mathbf{S}^{m-k} + c^{(2)} \mathbf{Q}_k \right)_{vu}.$$

- Assume $c_\sigma \leq 1$.
- We say that the pair (m, w) represents the **capacity** of the MPNN.

A measure of over-squashing

Definition

Given an MPNN with capacity (m, w), we define the **over-squashing** of v, u as

$$\mathsf{OSQ}_{v,u}(m, \mathsf{w}) := \Big(\sum_{k=0}^{m-1} \mathsf{w}^{2m-k-1} \Big(\mathsf{w}(\mathbf{S}^{m-k})^\top \mathrm{diag}(\mathbf{1}^\top \mathbf{S}^k)\mathbf{S}^{m-k} + c^{(2)}\mathbf{Q}_k \Big)_{vu} \Big)^{-1}.$$

- If $\mathsf{w} = \infty$ (infinite capacity), we have $\mathsf{OSQ}_{v,u}(m, \infty) = 0$.
- OSQ captures the case of *under-reaching*.
- If an MPNN learns a function y_{G} with mixing $\mathsf{mix}_{y_{\mathsf{G}}}(v, u)$, then

$$\mathsf{OSQ}_{v,u}(m, \mathsf{w}) < \left(\mathsf{mix}_{y_{\mathsf{G}}}(v, u)\right)^{-1}.$$

$\rightarrow$ Given capacity (m, w), **over-squashing occurs depending on the mixing required**

The depth required to induce enough mixing

- $\mathsf{A} = \mathrm{D}^{-1/2}\mathrm{A}\mathrm{D}^{-1/2}$
- $\boldsymbol{\Delta} := \mathrm{I} - \mathsf{A}$ is the **graph Laplacian** with eigenvalues $0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{n-1}$
- $\tau(v,u)$ is the commute time of nodes v, u
- $\gamma := \sqrt{d_{\max}/d_{\min}}$, with $d_{\max}$ and $d_{\min}$ the maximal and minimal degrees, respectively

Theorem

Assume $\max\{\mathsf{w}, \omega/\mathsf{w} + c_1\gamma + c_2\} \leq 1$. *If MPNN generates mixing* $\mathsf{mix}_{y_\mathsf{G}}(v,u)$ *among the features associated with nodes* v, u, *then the number of layers* m *satisfies*

$$m \geq \frac{\tau(v,u)}{4c_2} + \frac{|\mathsf{E}|}{\sqrt{d_v d_u}}\Big(\frac{\mathsf{mix}_{y_\mathsf{G}}(v,u)}{\gamma\mu} - \frac{1}{c_2}\Big(\frac{\gamma + |1 - c_2\lambda^*|^{r-1}}{\lambda_1} + 2\frac{c^{(2)}}{\mu}\Big)\Big),$$

where $r = d_\mathsf{G}(v,u)$, $\mu = 1 + 2c^{(2)}(1+\gamma)$ *and* $|1 - c_2\lambda^*| = \max_{0<\ell\leq n-1}|1 - c_2\lambda_\ell| < 1$.

How over-squashing hinders the expressive power of MPNNs

Necessary condition on the depth to induce enough mixing among nodes v, u

$\rightarrow$ Practical impossibility statement, since the commute time τ can be as large as $\mathcal{O}(n^3)$

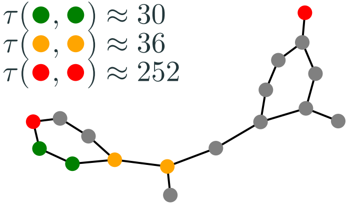
Corollary

On a graph with features, MPNNs as in the Theorem above with depth $m \leq n$, cannot learn functions that induce high mixing among features of nodes with large commute time.

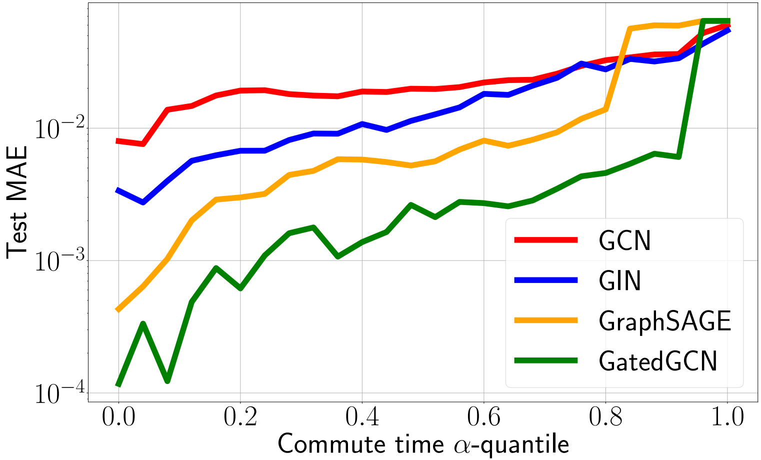
Our analysis identifies classes of functions that are harder to learn for an MPNNs, meaning that the capacity has to scale impractically with the graph-size

Synthetic ZINC

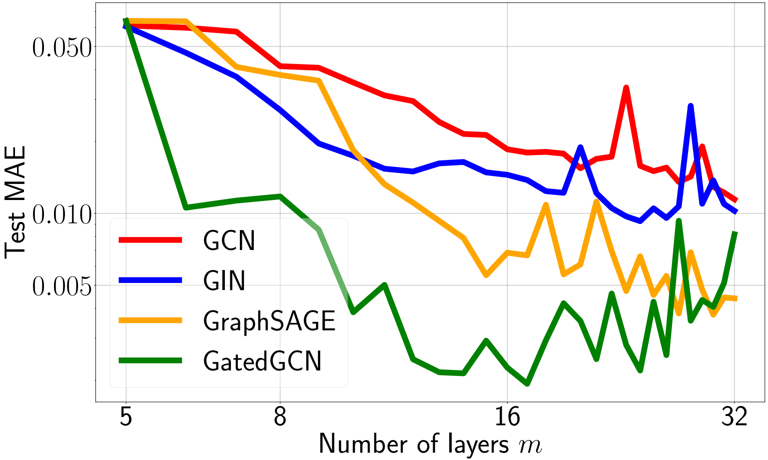
- $\{\mathsf{G}^i\}$ is the ZINC molecular graphs
- $x_v^i = 0$, except for two, which are set to uniform random numbers $x^i_{u^i}, x^i_{v^i}$ in $(0, 1)$
- Regression output is $y^i = \tanh(x^i_{u^i} + x^i_{v^i})$
- The two non-zero node features $x^i_{u^i}, x^i_{v^i}$ are positioned on G_i according to the α-quantile of the commute time τ distribution



The role of the commute time



The role of depth



On the level of mixing

Mixing	*input interval*	*maximal mixing*	GCN	GIN	GraphSAGE	GatedGCN
$\tanh(x^i_{u^i} + x^i_{v^i})$	$(0, 1)$	≈ 0.77	0.024	0.014	0.006	0.004
$\exp(x^i_{u^i} + x^i_{v^i})$	$(0, 1)$	≈ 7.4	0.043	0.021	0.033	0.008
$\exp(x^i_{u^i} + x^i_{v^i})$	$(0, 1.5)$	≈ 20.1	0.054	0.035	0.075	0.014

DRew: dynamically rewired message passing with delay

Some limitations of 'static' graph-rewiring techniques

Add edges among each pair of nodes within a certain distance (Graph-Transformers) (Abboud et al., 2022; Brüel-Gabrielsson et al., 2022; Ying et al., 2021; Rampášek et al., 2022) $\rightsquigarrow$ The new graph $\mathcal{R}(\mathsf{G})$ is much denser than G

- More sensitive to over-smoothing
- Higher impact on training time
- Loses inductive bias afforded by the distance

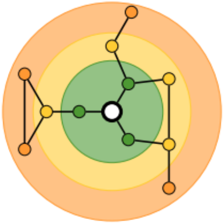
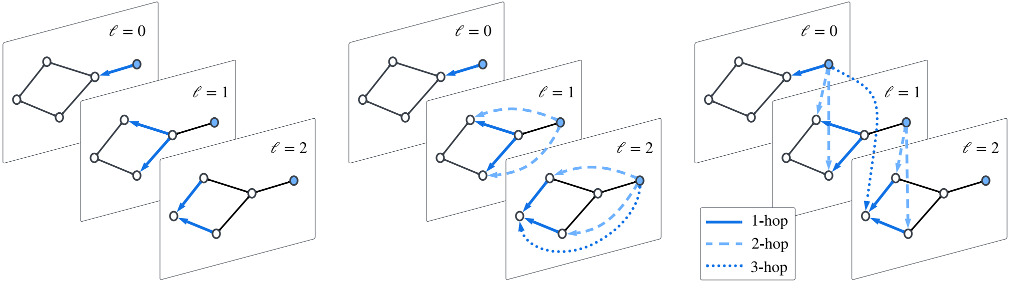


Figure 1: Figure from Abboud et al. (2022)

Dynamic edge-addition and delay: the new DREW and νDREW frameworks



(a) Classical MPNN (b) DREW – MPNN (c) νDREW – MPNN

- For classic MPNNs, information only travels from a node to its neighbours
- In DREW, at layer $r - 1$ we add edges connecting nodes at distance r
- In ν-DREW we introduce **delay depending on the distance between nodes**

ν-DREW in equations

Introduce $\tau_\nu(k) = \max(0, k - \nu)$ and consider the family of νDREW-MPNN

$$\mathbf{a}_{v,k}^{(t-1)} = \mathsf{agg}_k^{(t)} \left(\{\mathbf{h}_u^{(t-1-\tau_\nu(k))} : u \in \mathcal{N}_k(v)\}\right), 1 \leq k \leq t$$
$$\mathbf{h}_v^{(t)} = \mathsf{com}_k^{(t)} \left(\mathbf{h}_v^{(t-1)}, \mathbf{a}_{v,1}^{(t-1)}, \ldots, \mathbf{a}_{v,t}^{(t-1)}\right).$$

- The no delay case corresponds to $\nu = \infty$
- The agg and com are parameterised using weights depending on the layer t and the distance k

The advantages of νDREW

- The graph is progressively filled with each layer making for a more efficient framework

The advantages of νDREW

- The graph is progressively filled with each layer making for a more efficient framework
- We preserve the inductive bias provided by the input graph, since nodes that are closer still interact earlier

The advantages of νDREW

- The graph is progressively filled with each layer making for a more efficient framework
- We preserve the inductive bias provided by the input graph, since nodes that are closer still interact earlier
- The delay mechanism allow messages to travel both horizontally (i.e. across a layer) and vertically (i.e. across multiple layers) $\rightarrow$ we are 'rewiring' the graph by adding skip connections among different nodes based on their distance.

The advantages of νDREW

- The graph is progressively filled with each layer making for a more efficient framework
- We preserve the inductive bias provided by the input graph, since nodes that are closer still interact earlier
- The delay mechanism allow messages to travel both horizontally (i.e. across a layer) and vertically (i.e. across multiple layers) $\rightarrow$ we are 'rewiring' the graph by adding skip connections among different nodes based on their distance.
- Over-squashing is alleviated since now distant nodes can exchange messages directly

The advantages of νDREW

- The graph is progressively filled with each layer making for a more efficient framework
- We preserve the inductive bias provided by the input graph, since nodes that are closer still interact earlier
- The delay mechanism allow messages to travel both horizontally (i.e. across a layer) and vertically (i.e. across multiple layers) $\rightarrow$ we are 'rewiring' the graph by adding skip connections among different nodes based on their distance.
- Over-squashing is alleviated since now distant nodes can exchange messages directly
- We mitigate over-smoothing: a larger delay means that v aggregates the features from u before they are (significantly) 'smoothed' by repeated message passing

In Dwivedi et al. (2022) new graph benchmarks have been introduced, with tasks likely to exhibit long-range dependencies

- Graph-Transformers were shown to be superior to standard $\mathsf{MPNN}s$ across all tasks
- The message was: we need Graph-Transformers whenever the task has long-range interactions

Can we beat Graph-Transformers on these tasks using the $\nu\mathsf{DREW}$ formalism?

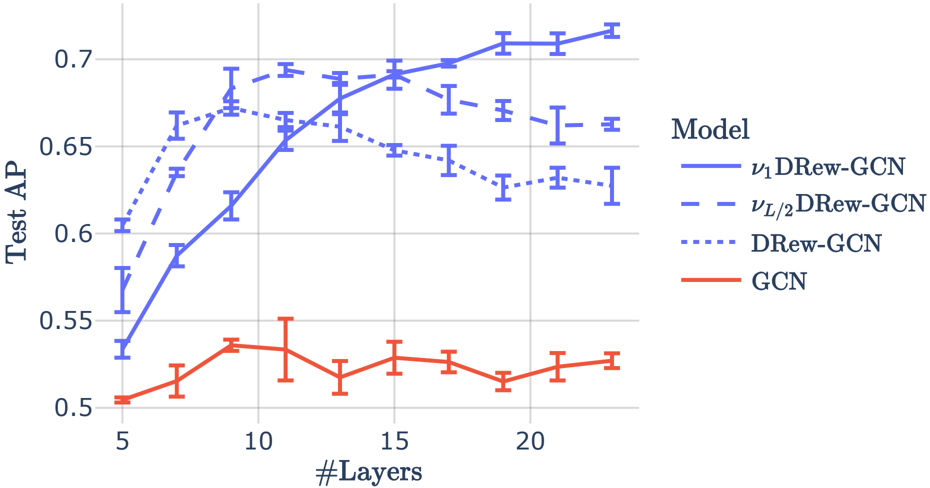
Experiments

Model	`Pept-func` AP ↑	`Pept-struct` MAE ↓	`PCQM-Contact` MRR ↑	`PascalVOC-SP` F1 ↑
GCN	0.5930±0.0023	0.3496±0.0013	0.3234±0.0006	0.1268±0.0060
+DRew	**0.6996±0.0076**	**0.2781±0.0028**	**0.3444±0.0017**	**0.1848±0.0107**
GINE	0.5498±0.0079	0.3547±0.0045	0.3180±0.0027	0.1265±0.0076
+DRew	**0.6940±0.0074**	**0.2882±0.0025**	**0.3300±0.0007**	**0.2719±0.0043**
GatedGCN	0.5864±0.0077	0.3420±0.0013	0.3218±0.0011	0.2873±0.0219
+DRew	**0.6733±0.0094**	**0.2699±0.0018**	**0.3293±0.0005**	**0.3214±0.0021**

Experiments

Model	Pept-func AP ↑	Pept-struct MAE ↓	Contact MRR ↑	PascalVOC-SP F1 ↑
DIGL+LapPE	0.6830±0.0026	**0.2616±0.0018**	0.1707±0.0021	0.2921±0.0038
MixHop+LapPE	0.6843±0.0049	**0.2614±0.0023**	0.3250±0.0010	0.2218±0.0174
Transf+LapPE	0.6326±0.0126	**0.2529±0.0016**	0.3174±0.0020	0.2694±0.0098
SAN+LapPE	0.6384±0.0121	0.2683±0.0043	**0.3350±0.0003**	**0.3230±0.0039**
GPS+LapPE	0.6535±0.0041	**0.2500±0.0005**	0.3337±0.0006	**0.3748±0.0109**
DRew-GCN	**0.6996±0.0076**	0.2781±0.0028	**0.3444±0.0017**	0.1848±0.0107
DRew-GCN+LapPE	**0.7150±0.0044**	**0.2536±0.0015**	**0.3442±0.0006**	0.1851±0.0092
DRew-GIN+LapPE	**0.7126±0.0045**	**0.2606±0.0014**	**0.3403±0.0035**	0.2692±0.0059
DRew-Gated+LapPE	**0.6977±0.0026**	**0.2539±0.0007**	0.3324±0.0014	**0.3314±0.0024**

The framework works over deep models



The framework works: classical regression tasks on QM9

Property	R-GIN+FA	R-GatedGNN+FA	SPN	DRew-GIN	$_1$DRew-GIN
mu	2.54±0.09	3.53±0.13	2.32±0.28	**1.93±0.06**	2.00±0.05
alpha	2.28±0.04	2.72±0.12	1.77±0.09	**1.63±0.03**	**1.63±0.05**
HOMO	1.26±0.02	1.45±0.04	1.26±0.09	**1.16±0.01**	**1.17±0.02**
LUMO	1.34±0.04	1.63±0.06	1.19±0.05	**1.13±0.02**	**1.15±0.02**
gap	1.96±0.04	2.30±0.05	1.89±0.11	**1.74±0.02**	**1.74±0.03**
R2	12.61±0.37	14.33±0.47	10.66±0.40	**9.39±0.13**	9.94±0.07
ZPVE	5.03±0.36	5.24±0.30	**2.77±0.17**	**2.73±0.19**	**2.90±0.30**
U0	2.21±0.12	3.35±1.68	1.12±0.13	**1.01±0.09**	**1.00±0.07**
U	2.32±0.18	2.49±0.34	**1.03±0.09**	**0.99±0.08**	**0.97±0.04**
H	2.26±0.19	2.31±0.15	**1.05±0.04**	**1.06±0.09**	**1.02±0.09**
G	2.04±0.24	2.17±0.29	**0.97±0.06**	1.06±0.14	**1.01±0.05**
Cv	1.86±0.03	2.25±0.20	1.36±0.06	**1.24±0.02**	**1.25±0.03**

Thank you!

For any question do not hesitate to contact me:

Email : fd405 at cam.ac.uk

Social : *@Francesco_dgv*

References

Abboud, R., Dimitrov, R., and Ceylan, İ. İ. (2022). Shortest path networks for graph property prediction. *arXiv preprint arXiv:2206.01003*.

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

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. (2022). Long range graph benchmark. *arXiv preprint arXiv:2206.08164*.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022). Recipe for a general, powerful, scalable graph transformer. *arXiv preprint arXiv:2205.12454*.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *ICLR*. OpenReview.net.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021). Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.