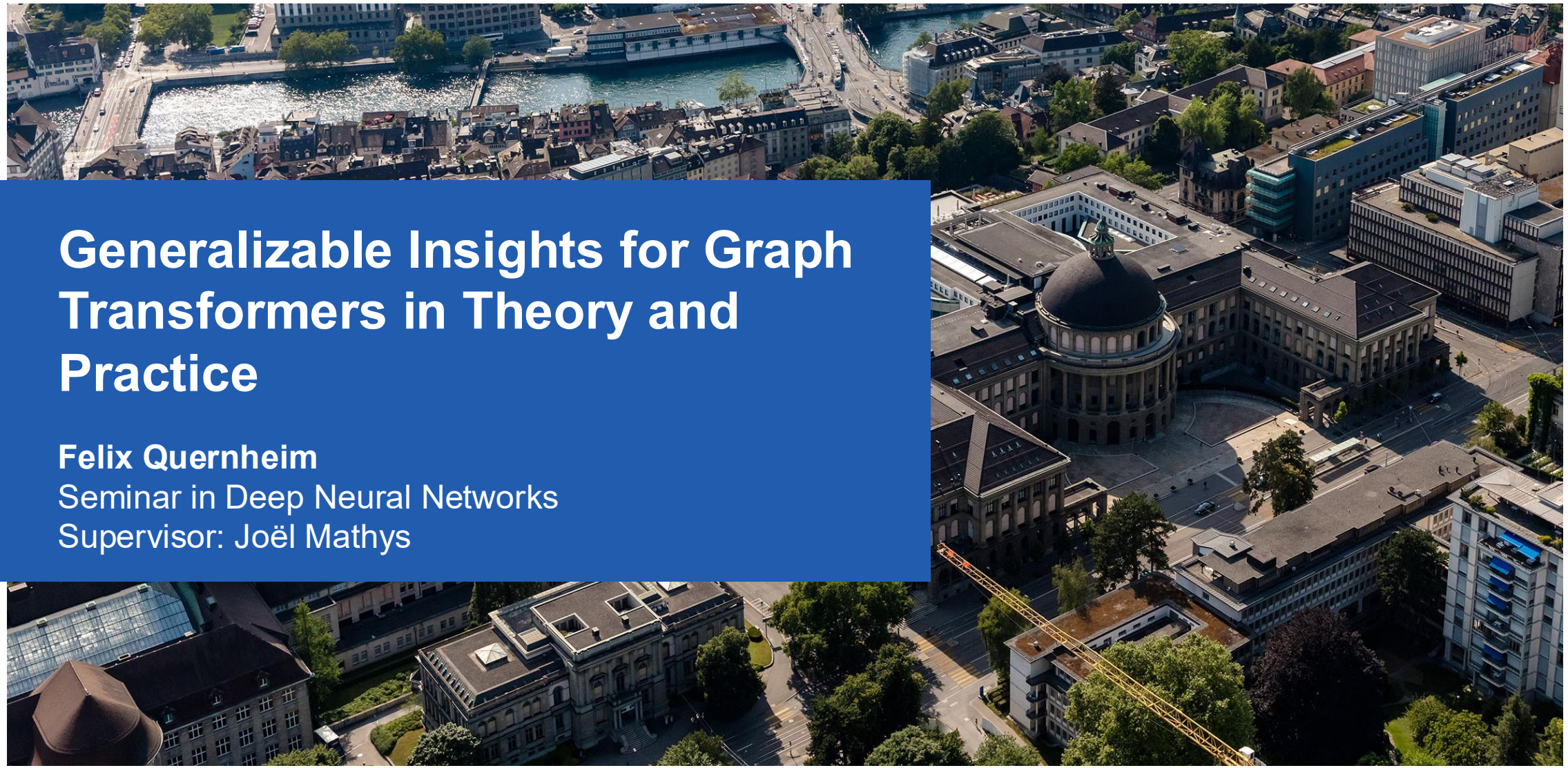
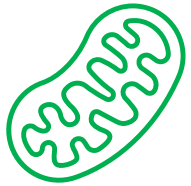


Generalizable Insights for Graph Transformers in Theory and Practice

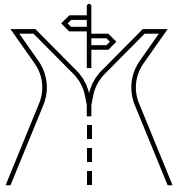
Felix Quernheim
Seminar in Deep Neural Networks
Supervisor: Joël Mathys

Graphs are everywhere

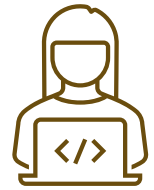


Biology

Social Networks



Transportation Networks



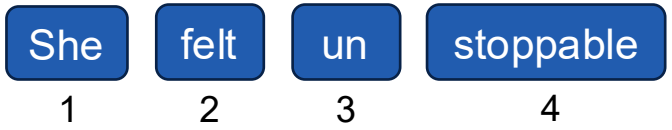
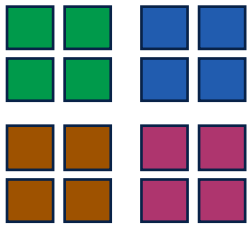
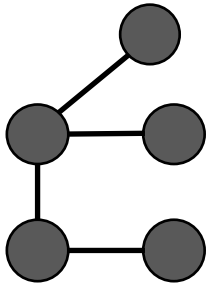
Coding

Why Transformers?

NLP	Computer Vision	Audio Processing
• Machine translation • Summarisation • …	• Image classification • Object detection • …	• Speech recognition • Music generation • …

one architecture, many applications

From Sequences to Graphs – What changes?

Text	Images	Graphs
	 Regular grid	 ?
Fixed left-to-right order → positions are natural	Regular 2D grid → Spatial neighbours are well-defined	No canonical ordering → node-positions must be learned or encoded

GraphTransformers – A Fragmented Landscape

3 obstacles in the current literature:

1. **Architecture-tied expressivity**
   → theoretical results only hold for custom architecture
2. **Graph-specific attention**
   → heavy deviations of suggested GTs from standard architecture
3. **Limited evaluation**
   → use of different and few benchmarks

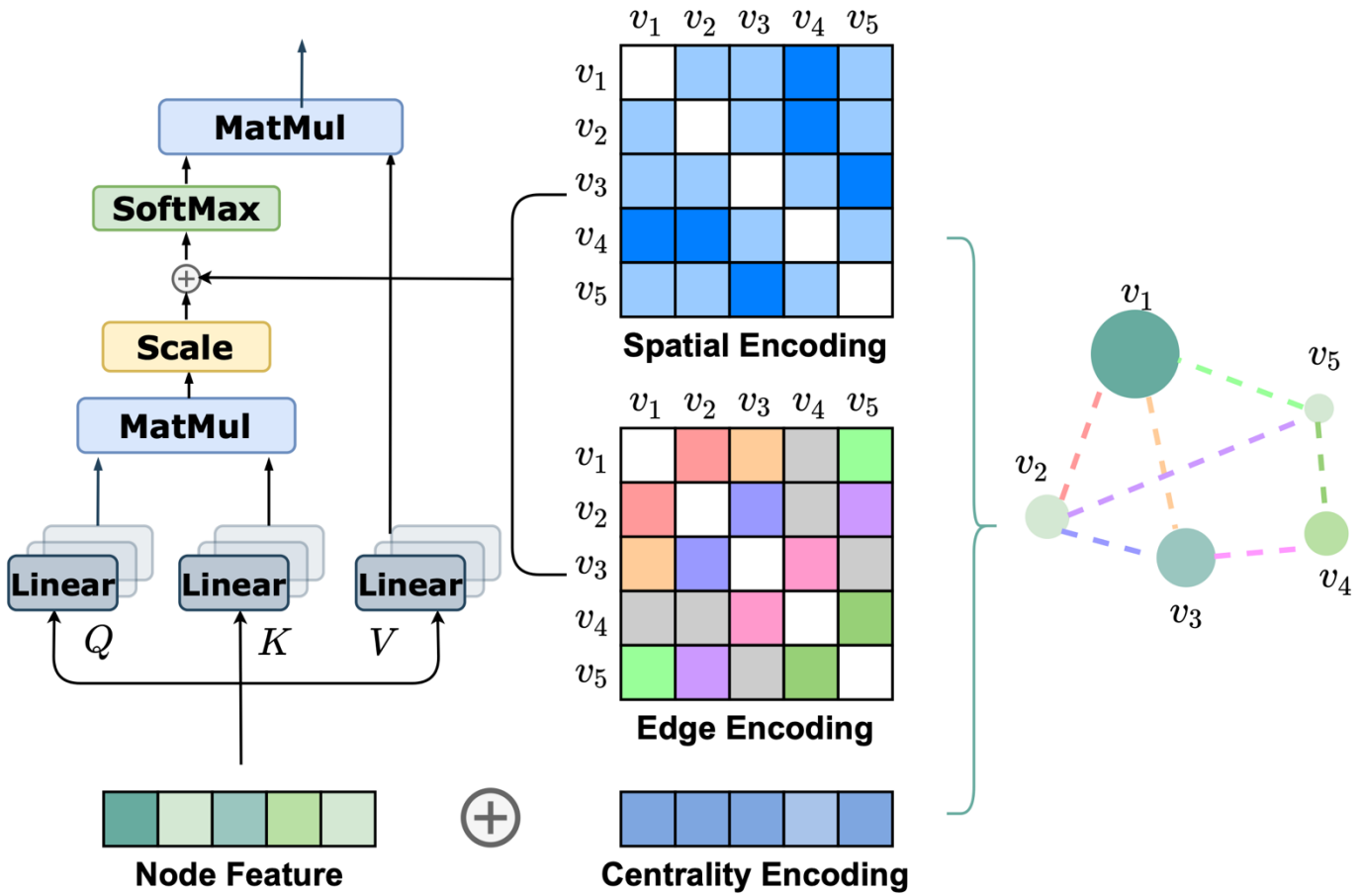
Related Literature

– Graph Transformers and Positional Encodings –

Example 1: Graphormer

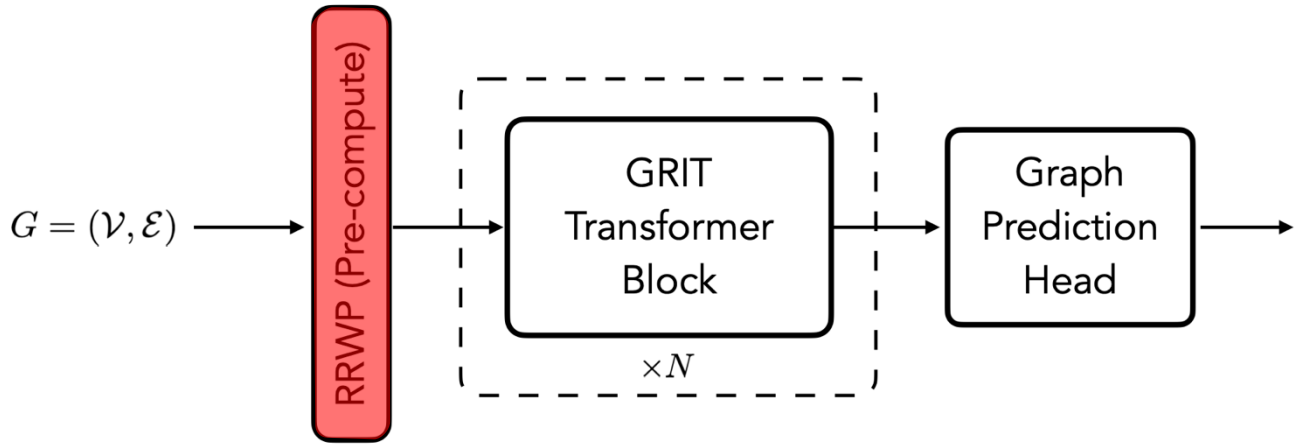
3 ”**hard coded**”, graph-specific biases added to standard attention:

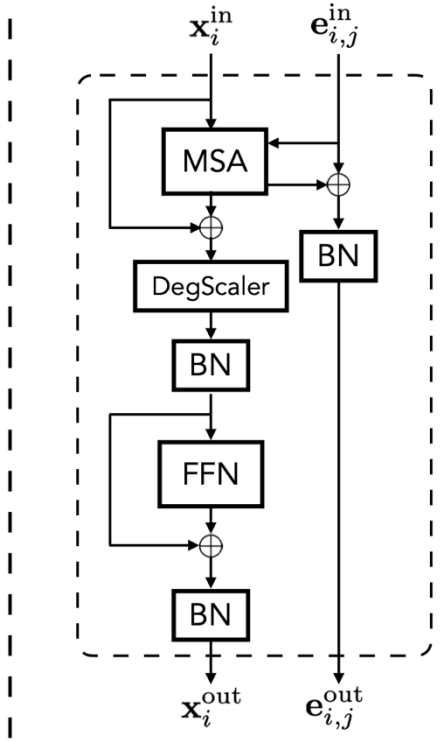
- degree (input),
- shortest-path distance (attention bias),
- edge features along shortest path (attention bias)



Architecture suggested and figure taken from *Ying et al., 2021*

Example 2: GRIT





$$\hat{\boldsymbol{e}}_{i,j} = \sigma\left(\rho\left(\left(\boldsymbol{W}_Q\,\boldsymbol{x}_i + \boldsymbol{W}_k\,\boldsymbol{x}_j\right) \odot \boldsymbol{W}_{Ew}\boldsymbol{e}_{i,j}\right) + \boldsymbol{W}_{Eb}\boldsymbol{e}_{i,j}\right) \in \mathbb{R}^{d'}$$

$$\alpha_{i,j} = Softmax_{j\in V}\left(\boldsymbol{W}_A\hat{\boldsymbol{e}}_{i,j}\right) \in \mathbb{R}$$

$$\hat{\boldsymbol{x}}_i = \sum_{j\in V} \alpha_{i,j} \cdot \left(\boldsymbol{W}_V\boldsymbol{x}_j + \boldsymbol{W}_{E_V}\hat{\boldsymbol{e}}_{i,j}\right)$$

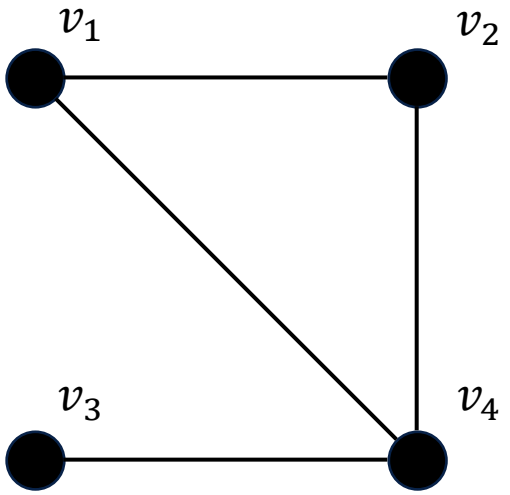
Architecture suggested and figure taken from *Ma et al., 2023*

Positional Encoding via Random Walks

Ingredients:

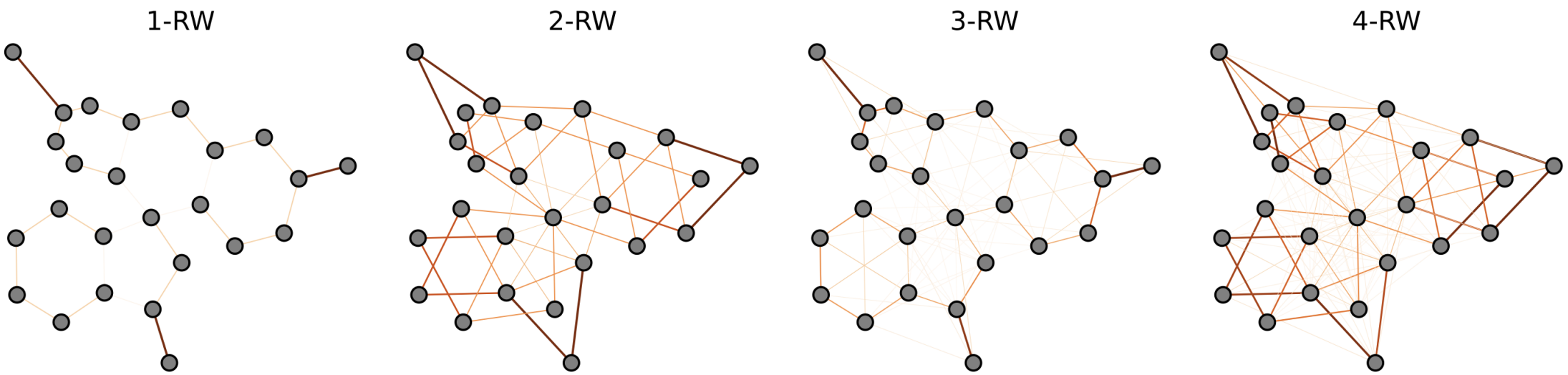
- Adjacency matrix: $\boldsymbol{A} \in \mathbb{R}^{n \times n}$
- Degree matrix: $\mathbf{D} \in \mathbb{R}^{n \times n}$
- We set: $\boldsymbol{M} \coloneqq \boldsymbol{D^{-1}A}$

$$\boldsymbol{M} = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$



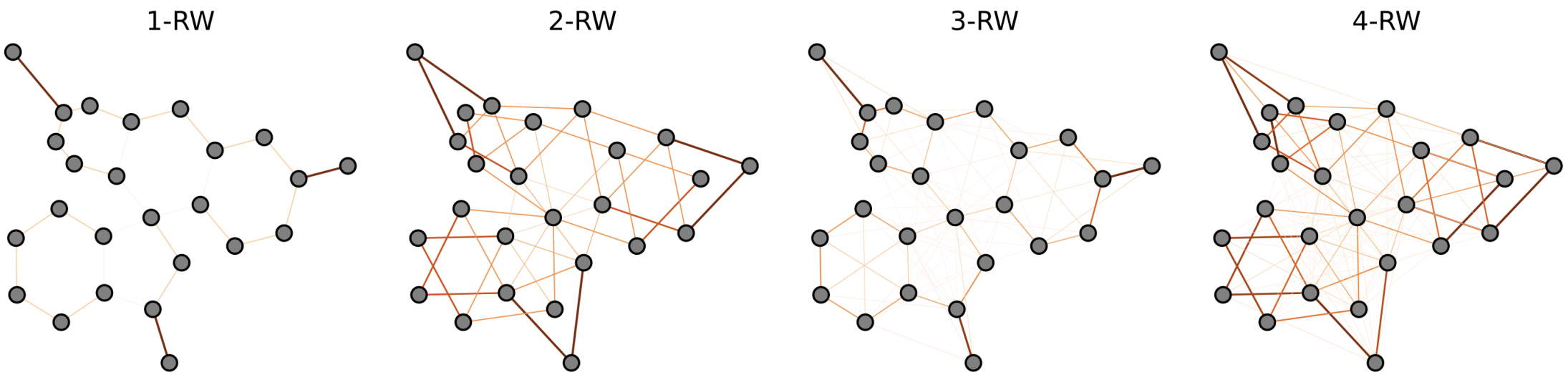
→ $\boldsymbol{M}_{i,j}$ captures probability to walk from i to j in one step

Positional Encoding via Random Walks



Random walks on the fluorescein molecule, up to the 4th power. Thicker and darker edges indicate higher edge weight. (Ma et al., 2023)

Positional Encoding via Random Walks



RWSE:

→ track only **return probabilities**

→ diagonal entries of matrices in $\boldsymbol{P}$

RRWP:

→ track **all probabilities**

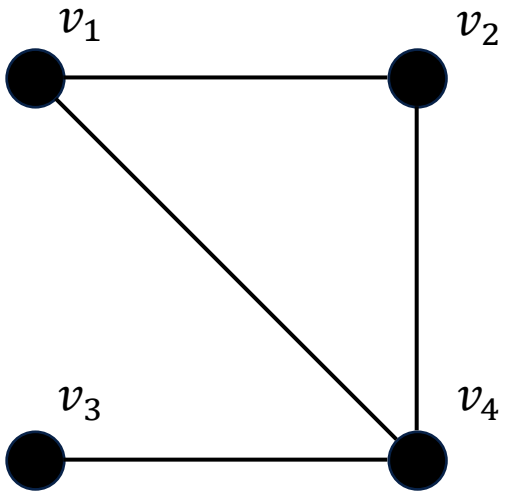
→ all entries of $\boldsymbol{P}$ needed

Positional Encoding via the Laplacian

Ingredients:

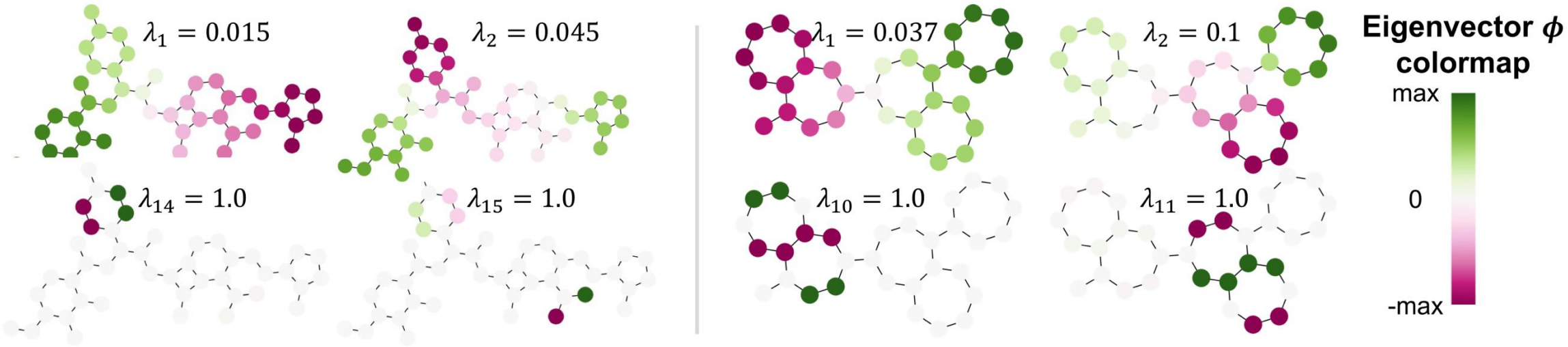
- Adjacency matrix: $\boldsymbol{A} \in \mathbb{R}^{n \times n}$
- Degree matrix: $\mathbf{D} \in \mathbb{R}^{n \times n}$
- We set: $\mathbf{L} \coloneqq \boldsymbol{D} - \boldsymbol{A}$

$$\boldsymbol{L} = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$



→ The eigendecomposition of $\boldsymbol{L}$ reveals structural information about the graph

Positional Encoding via the Laplacian



For any (normalized) eigenvector $\boldsymbol{v}_i$, eigenvalue λ_i pair, we have

$$\boldsymbol{v}_i^T \boldsymbol{L} \boldsymbol{v}_i = \lambda_i = \sum_{(u,w) \in E} (\boldsymbol{v}_i(u) - \boldsymbol{v}_i\,(w))^2$$

Figure taken from *Kreuzer et al., 2021*

Positional Encoding via the Laplacian

1. Compute k smallest eigenvalues and corresponding eigenvectors
2. Turn them into per-node spectral features

LPE:

→ use raw eigenvectors and eigenvalues

$$\underbrace{\phi\left(\mathbf{V}_i^T, \lambda + \varepsilon\right)}_{\in \mathbb{R}^{n \times k}}$$

SPE:

→ Use basis-invariant matrices

$$\underbrace{\mathbf{V}\ \mathrm{diag}(\phi_i(\lambda))\ \mathbf{V}^T}_{\in \mathbb{R}^{n \times n}}$$

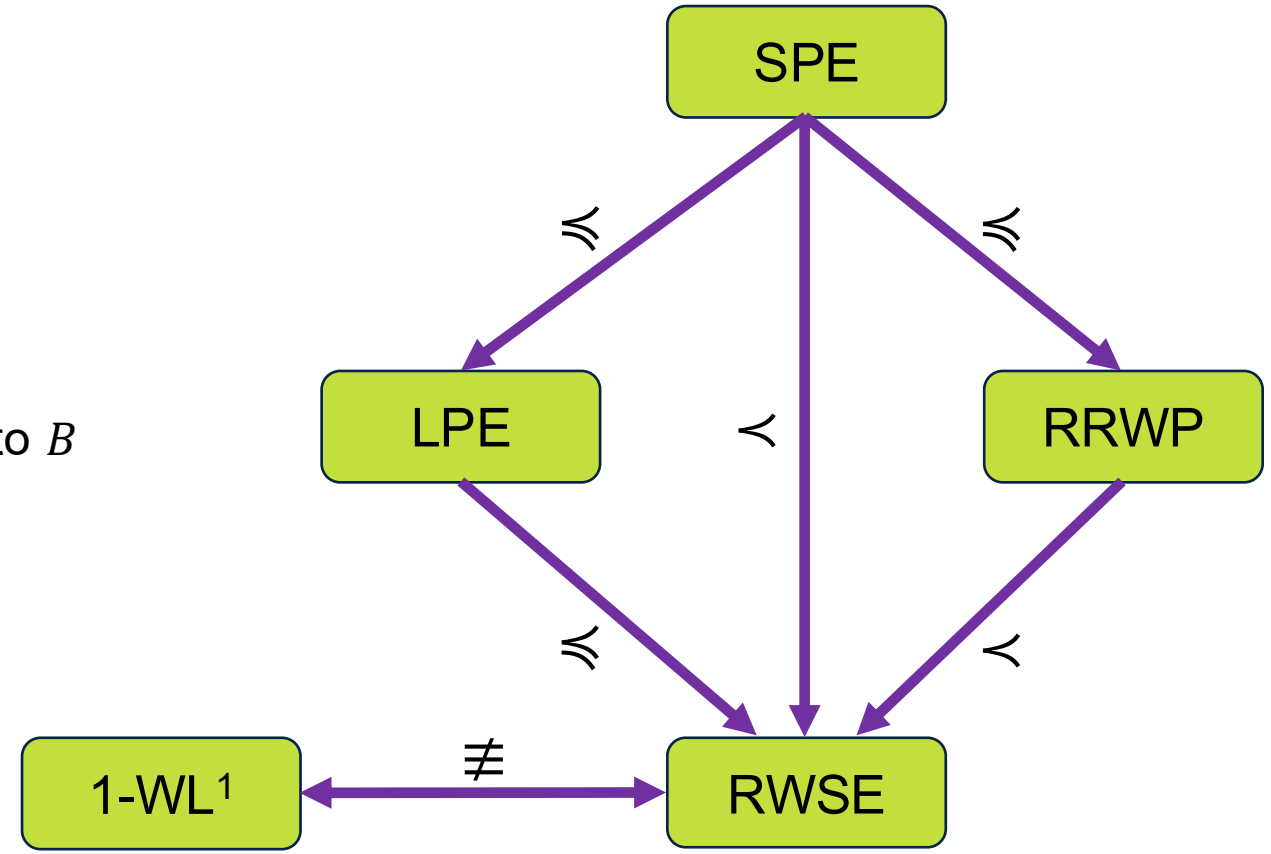
3. Aggregate with a permutation equivariant network

For SPE, see also *Huang et al., 2024*

Contribution of the Paper

Hierarchy of Positional Encodings

$A \prec B$ ($A \preccurlyeq B$, $A \not\equiv B$):
algorithm A is strictly more powerful
(at least as powerful, incomparable) than/to B



Excerpt of figure 2 in *Stoll et al., 2025*

[1]1-WL: Weisfeiler-Leman algorithm

The General Distance Transformer

Attention mechanism: For $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$ and $\mathbf{B} \in \mathbb{R}^{L \times L}$, biased attention is

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{B}) := \text{softmax}\left(d^{-\frac{1}{2}} \cdot \mathbf{Q}\mathbf{K}^T + \mathbf{B}\right)\mathbf{V}$$

→ The proposed architecture mainly resembles the standard transformer encoder

+ additional bias

Attention Bias and Positional Encodings

recall: $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{B}) := \text{softmax}\left(d^{-\frac{1}{2}} \cdot \mathbf{Q}\mathbf{K}^T + \mathbf{B}\right)\mathbf{V}$

Absolute PEs: RWSE, LPE, SPE

Relative PE: RRWP

$$\mathbf{X}_i := \ell_V(i)$$

$$\mathbf{B}_{ij} := \rho\left(\ell_E(i,j)\right)$$

$$\mathbf{X}_i := \ell_V(i) + \mathbf{P}_i\mathbf{W}_p$$

$$\mathbf{B}_{ij} := \rho\left(\ell_E(i,j)\right) + \mathbf{U}_{ij}\mathbf{W}_U$$

Experiments – Real-world Datasets

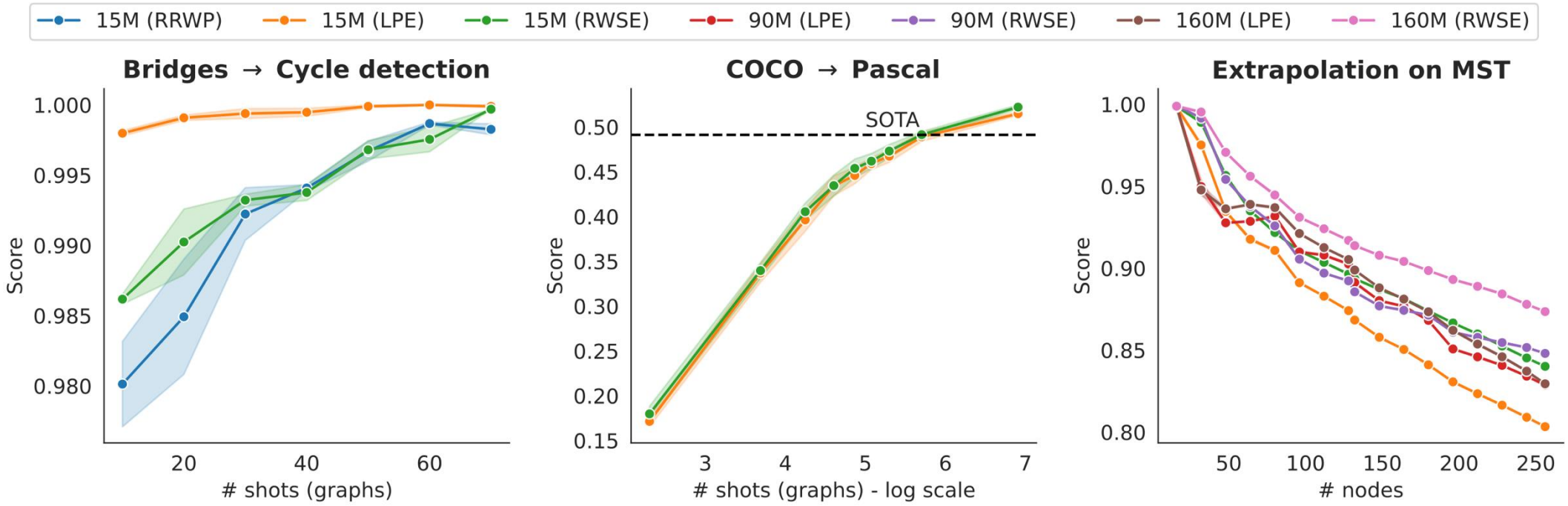
PE	Mean Rank	PCQ MAE ↓	COCO F1 ↑	CODE F1 ↑	FLOW MAE ↓	MST F1 ↑	BRIDGES F1 ↑
NoPE	3.50	93.6 ± 0.5	43.12 ± 00.85	19.27 ± 00.20	1.73 ± 0.09	93.29 ± 00.88	55.36 ± 24.94
LPE	2.50	92.7 ± 0.9	44.83 ± 00.71	19.48 ± 00.21	1.75 ± 0.12	91.08 ± 00.95	91.76 ± 07.66
SPE	4.00	94.1 ± 0.6	43.87 ± 00.54	19.35 ± 00.21	1.98 ± 0.14	92.52 ± 00.12	54.81 ± 21.20
RWSE	2.67	92.9 ± 0.6	43.82 ± 01.01	19.39 ± 00.47	1.49 ± 0.02	93.26 ± 00.45	87.34 ± 03.97
RRWP	**2.33**	90.4 ± 0.3	39.91 ± 01.07	19.42 ± 00.10	1.45 ± 0.06	96.04 ± 00.91	99.21 ± 00.09

→ NoPE = no Positional Encoding

→ Though SPE is theoretically the most powerful PE, it preforms worst on average

→ RRWP performs especially good on theoretical tasks

Experiments – Testing Generalizability



→ Few-shot transfer from Bridges to Cycles is trivial

→ Few-shot transfer works well on image dataset

→ Good results for extrapolation might be due easy Heuristics

Experiments – Comparison to Graphormer

Model	PE	Basic	Regular	Extension	CFI	Total
Graphormer-GD	RWSE	57	50	96	0	203
Graphormer-GD	LPE	55	40	85	3	183
GDT	RWSE	57	50	96	0	203
GDT	LPE	54	39	84	3	180

→ Results shown are for the BREC benchmark

→ Graphormer and GDT preform similar

Discussion

Strengths	Weaknesses
• Unified theoretical framwork • Benchmarks for various domains • Incorportation of various positional encodings	• Lacking support for the GDT architecture in FlashAttention • Limited baseline comparison • Lacking reflection of true (lack of) complexity of some experiments • Just one relative PE considered (RRWP)

Q & A

References

Huang, N. T. and Villar, S. (2021). A short tutorial on the Weisfeiler-Lehman test and its variants. In *ICASSP 2021 – 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8533–8537. IEEE.

Huang, Y., Lu, W., Robinson, J., Yang, Y., Zhang, M., Jegelka, S., and Li, P. (2024). On the stability of expressive positional encodings for graphs. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, P. K., Coates, M., Torr, P. H. S., and Lim, S.-N. (2023). Graph inductive biases in transformers without message passing. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 23321–23337. PMLR.

Stoll, T., Müller, L., and Morris, C. (2025). Generalizable insights for graph transformers in theory and practice. *arXiv preprint arXiv:2511.08028*.

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? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 28877–28888.

Backup Slide 1 – Weisfeiler Leman Graph Isomorphism Test

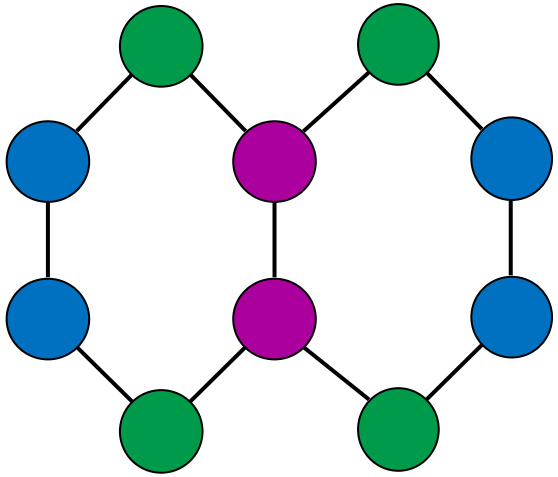
Algorithm *1-WL (color refinement)*

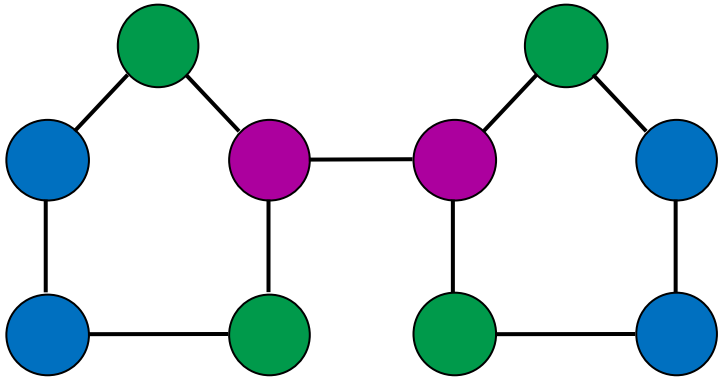
Input: $G = (V, E, X_V)$

1. $c_v^0 \leftarrow \text{hash}(X_v)$ for all $v \in V$
2. **repeat**
3. $\quad c_v^\ell \leftarrow \text{hash}(c_v^{\ell-1}, \{\!\{c_w^{\ell-1} : w \in \mathcal{N}_G(v)\}\!\})\ \forall v \in V$
4. **until** $(c_v^\ell)_{v \in V} = (c_v^{\ell-1})_{v \in V}$
5. **return** $\{\!\{c_v^\ell : v \in V\}\!\}$

Pseudocode from *Huang et al., 2021*

Backup Slide 2 – 1-WL-algorithm – Indistinguishable Graphs





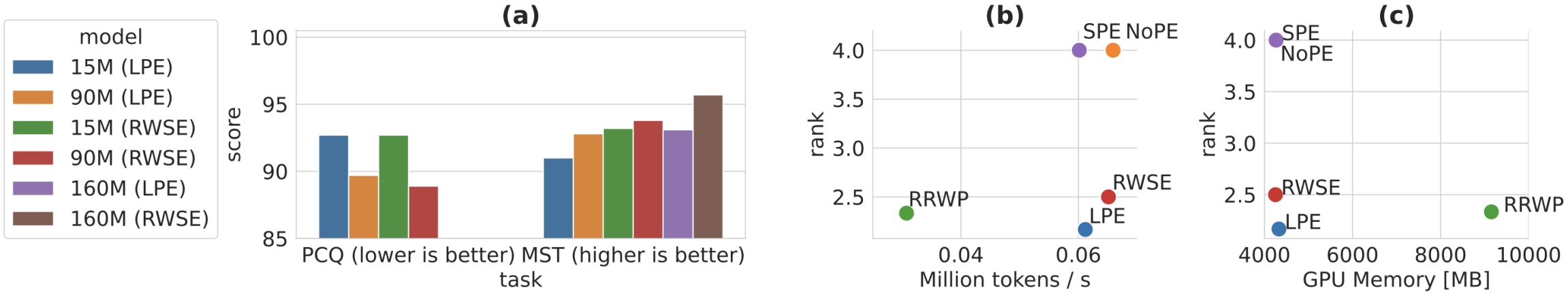
Backup Slide 3 – Definitions of Evaluation Metrics

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Backup Slide 4 – Different Model Sizes and Training Speed



(a): Results on 90M and 160M models for PCQ and MST evaluated on LPE and RWSE

(b): Number of tokens evaluated per second during training for each PE.
Results are obtained by averaging runtimes per token across tasks

(c): Average GPU memory requirement for each PE