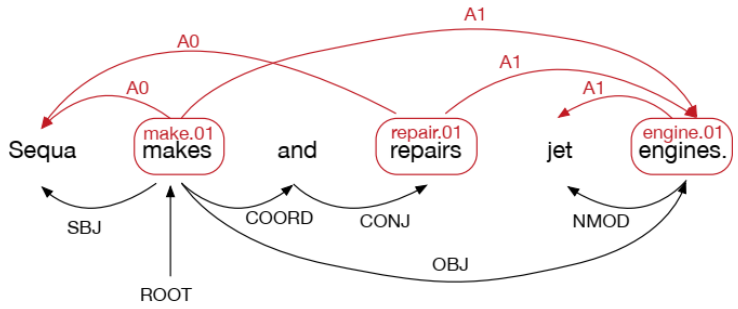
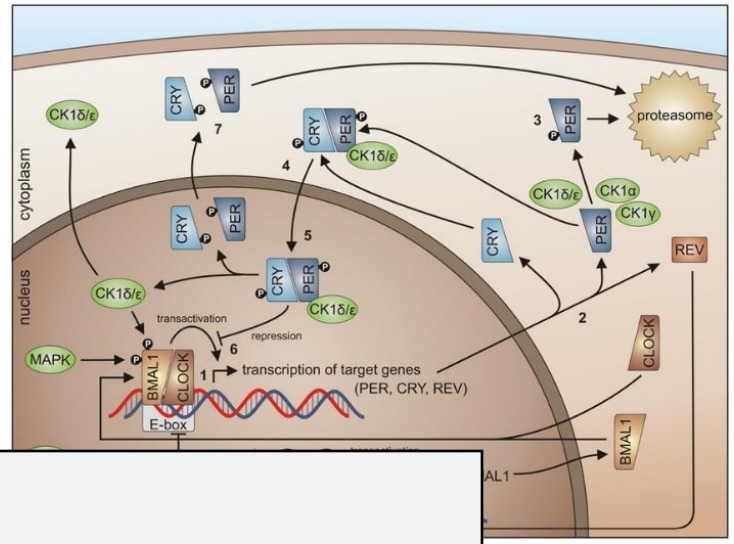
Graph Neural Networks: Architectures

Seminar in Deep Neural Networks, 27.04.2021

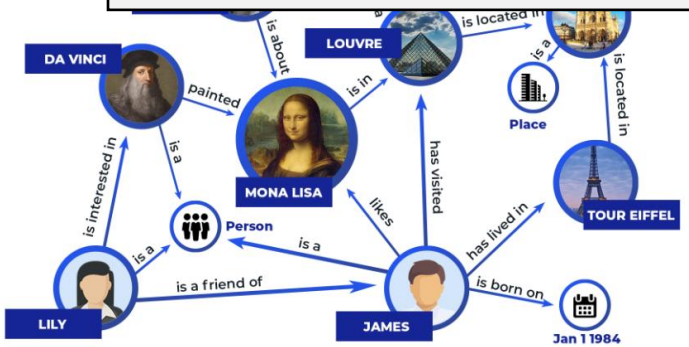
Susanne Keller

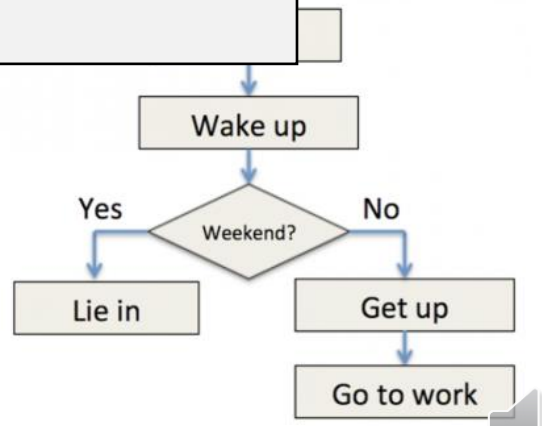
1. Why Graph Neural Networks?





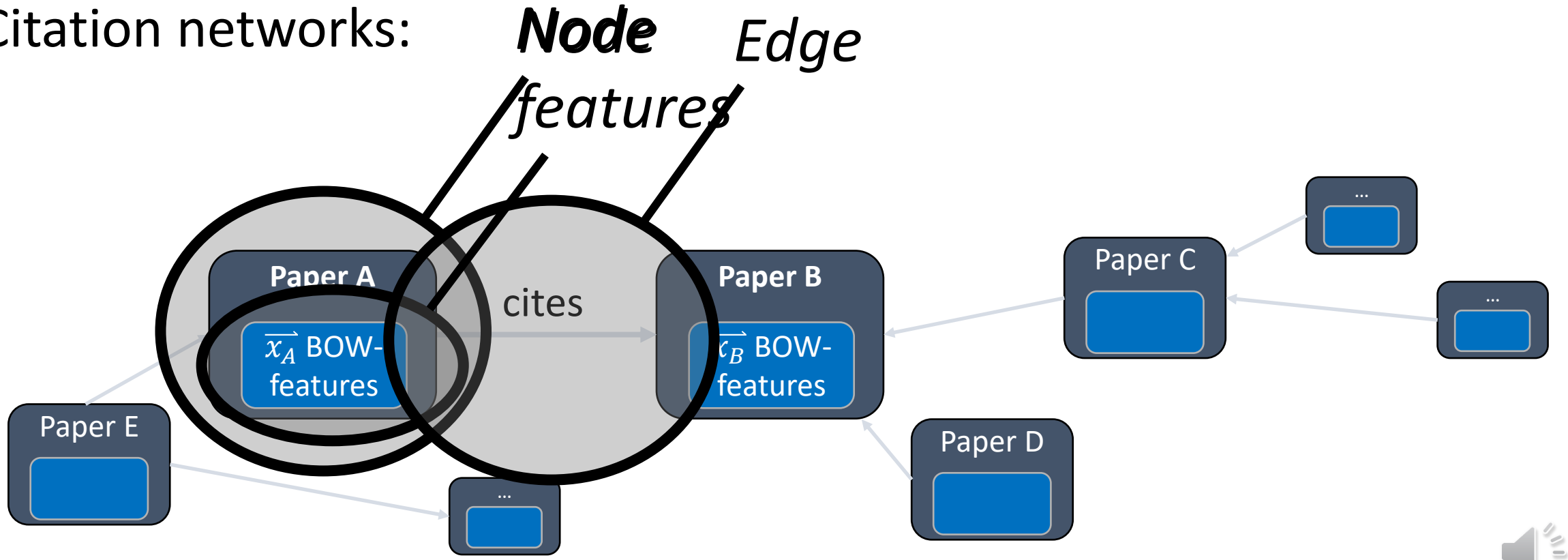
Graph-structured data!





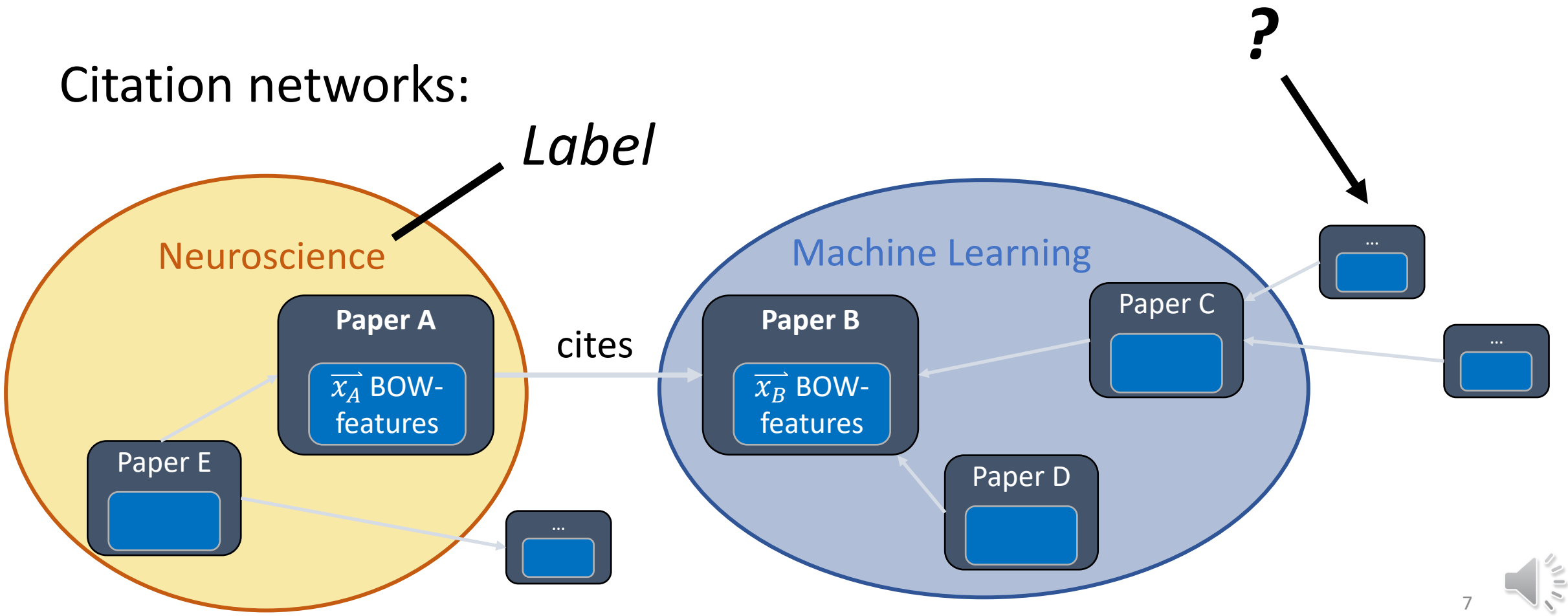
Historical example: Finding a paper's topic

Citation networks:

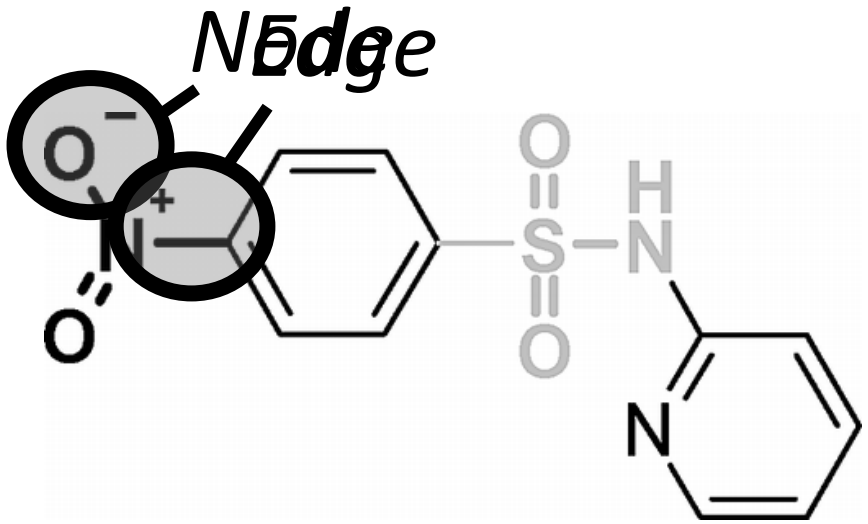


Historical example: Finding a paper's topic

Citation networks:

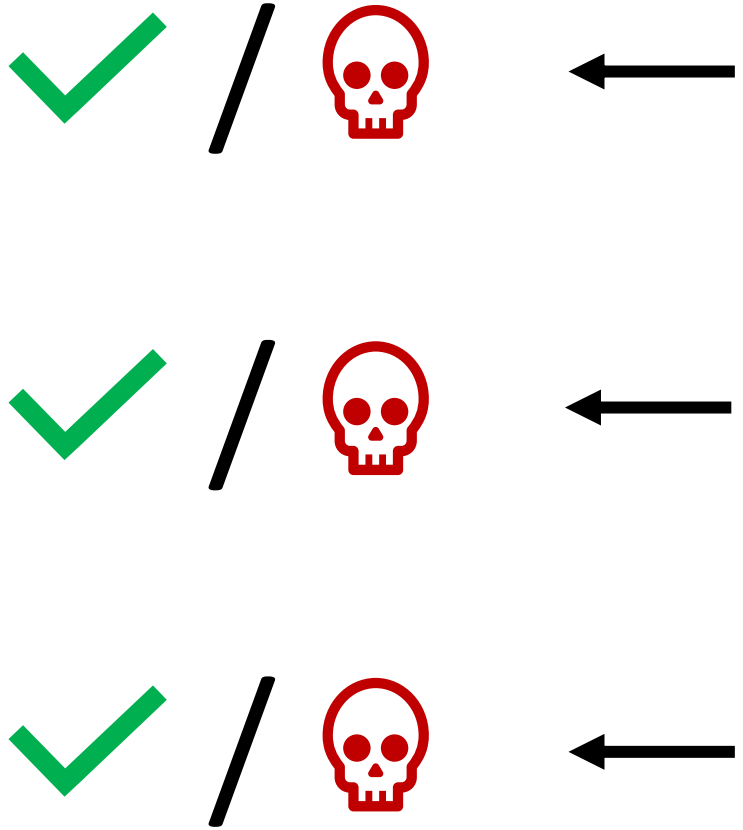


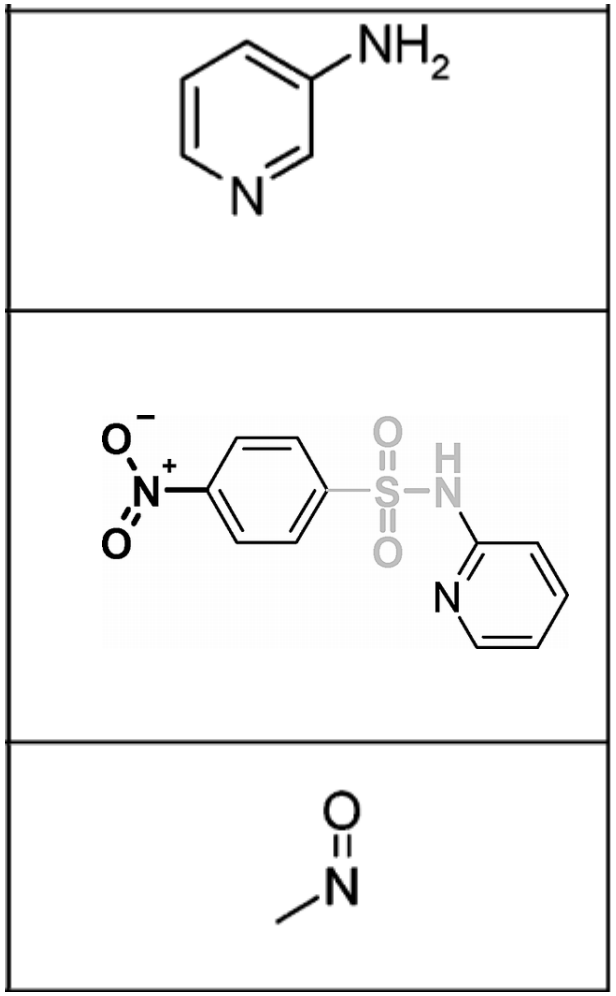
Example: Mutagenic or not?



[D4] https://pubs.acs.org/doi/abs/10.1021/jm040835a

Example: Mutagenic or not?

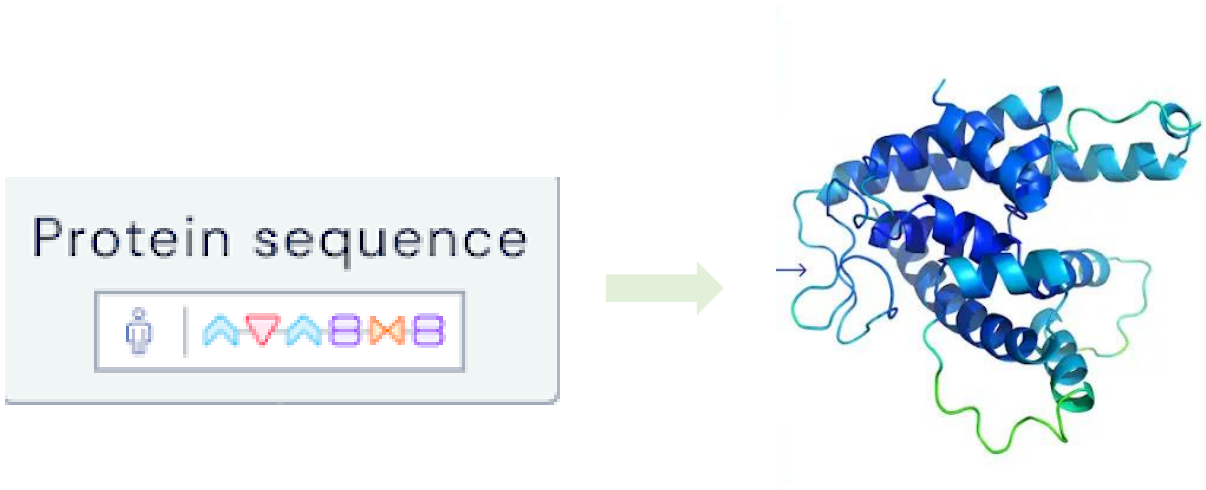




https://pubs.acs.org/doi/abs/10.1021/jm040835a

An important task with graph structure

- Protein structure prediction

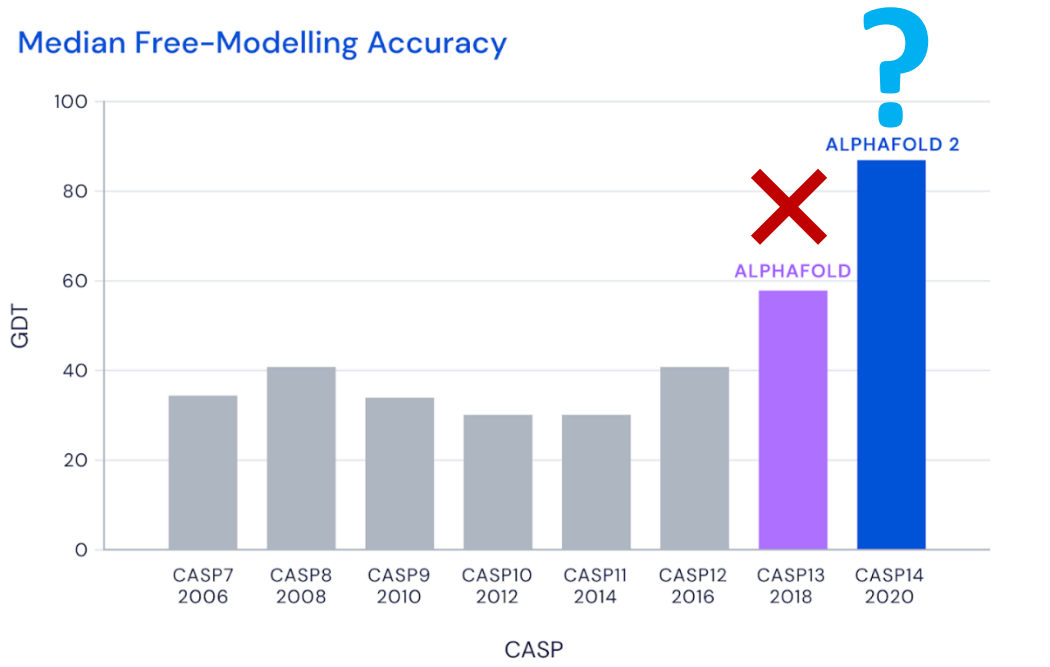


https://en.wikipedia.org/wiki/AlphaFold

An important task with graph structure: CASP

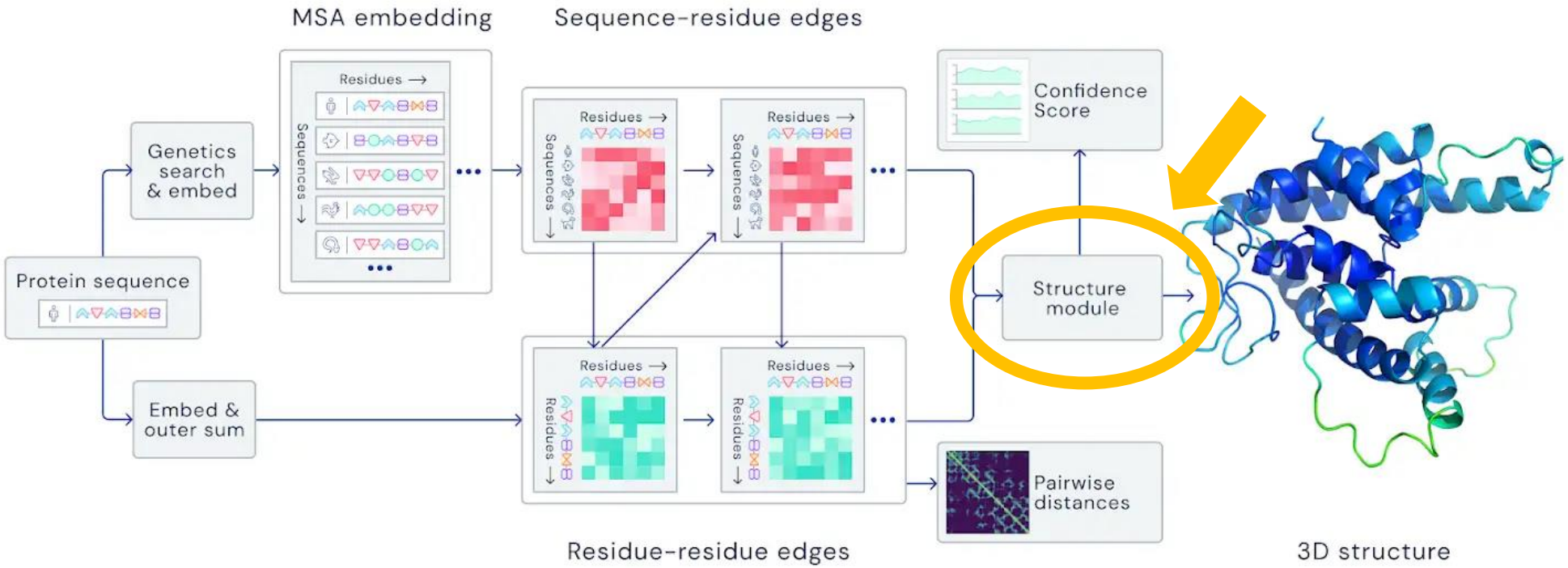
- Protein structure prediction
- Unsolved for 50 years
- Recent successes:

Graph neural networks?



https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

Example: **AlphaFold 2** (Google Deep Mind)



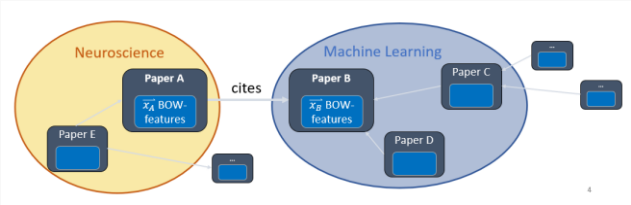
- Makes use of a ‘structural module’ (but GNNs? We don’t know)

Task types:

Transfer to unseen nodes or edges

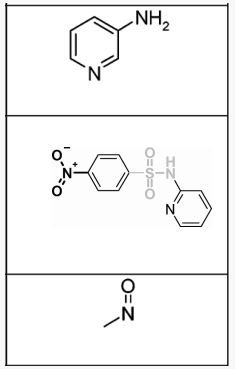
Single graphs

- semi-supervised

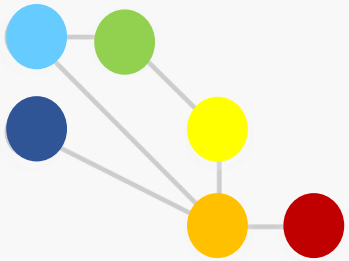


Many graphs

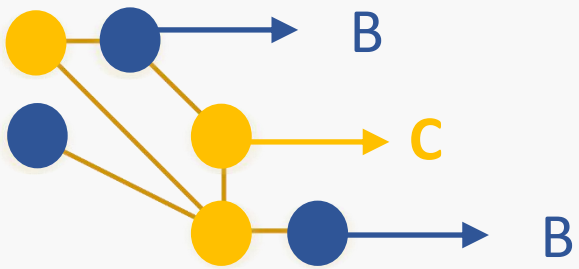
Transfer to unseen graphs



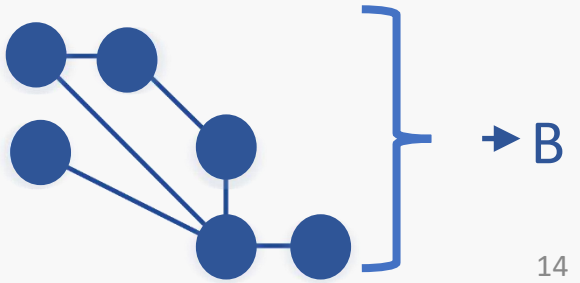
Graph generation



Node-level inference

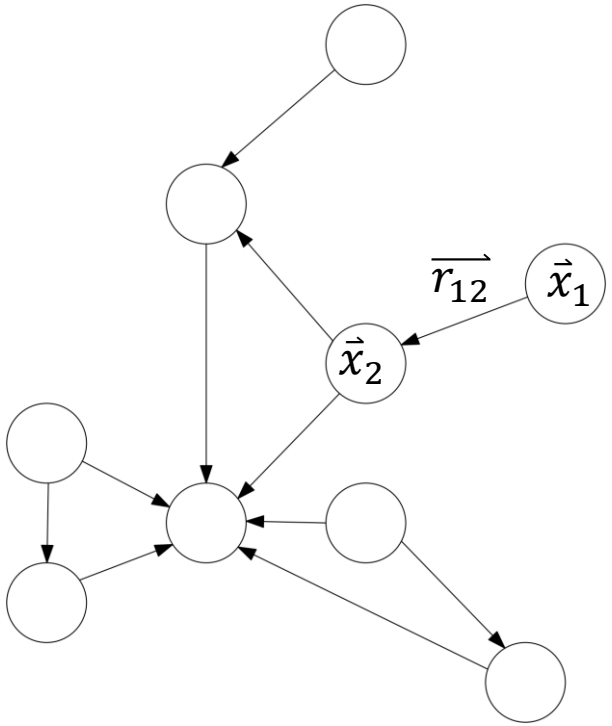


Graph classification, etc



Why not just use standard NNs?

- Adjacency matrix as input to a neural network?



Reminder: Graph G = (V, E)

V: Vertices, V = {1, 2, … n}
or V = {1: $\vec{x}_1$, 2: $\vec{x}_2$, … } ($\overrightarrow{x_i}$: features)

E: Edges, e.g. E = {(i, j) | ∃ edge **i → j** }
or E = {(i, j, $\overrightarrow{r_{ij}}$)} ($\overrightarrow{r_{ij}}$: edge features)

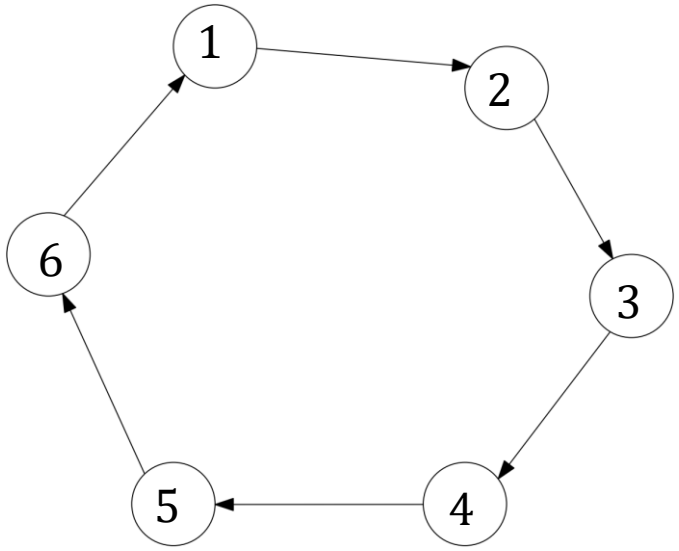
Adjacency matrix A for *undirected* graphs:

A_{ij} = 1 or 0; 1 if ∃ edge (i, j)

$N(i)$: Set of direct neighbours of node i

Why not just use standard NNs?

- Adjacency matrix as input to a neural network?

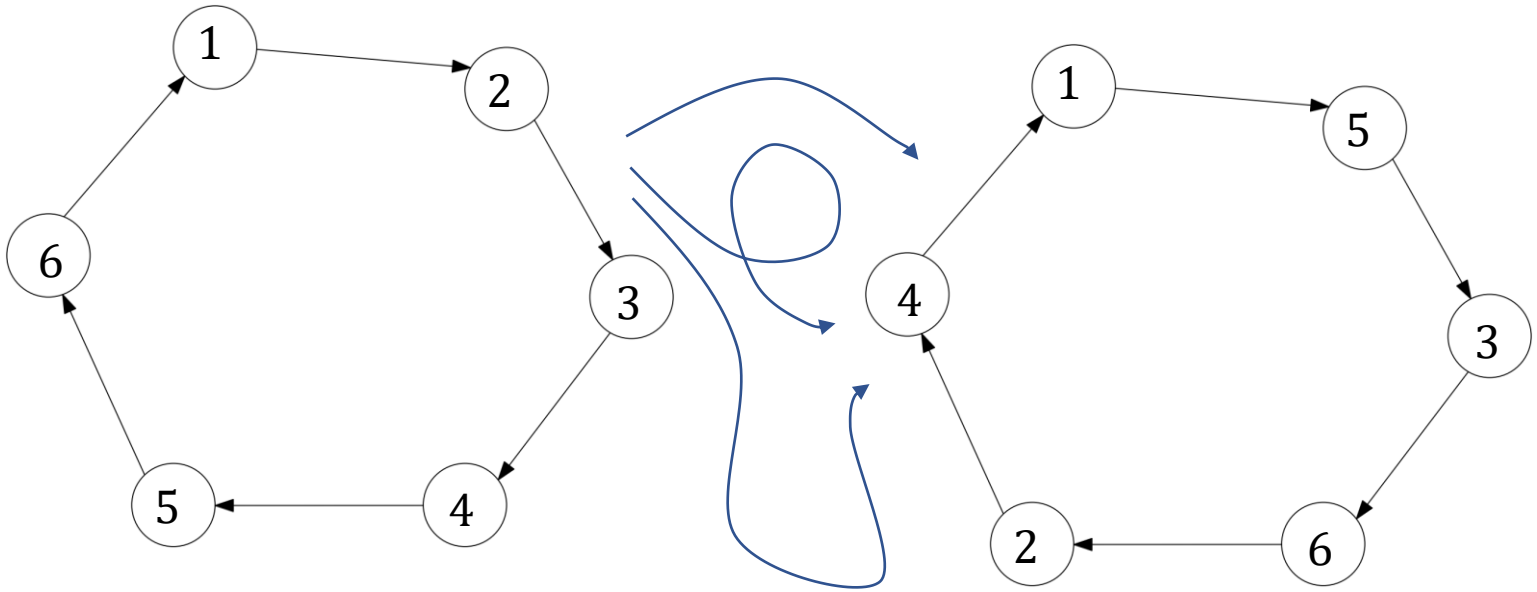


A

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Why not just use standard NNs?

- Adjacency matrix as input to a neural network?

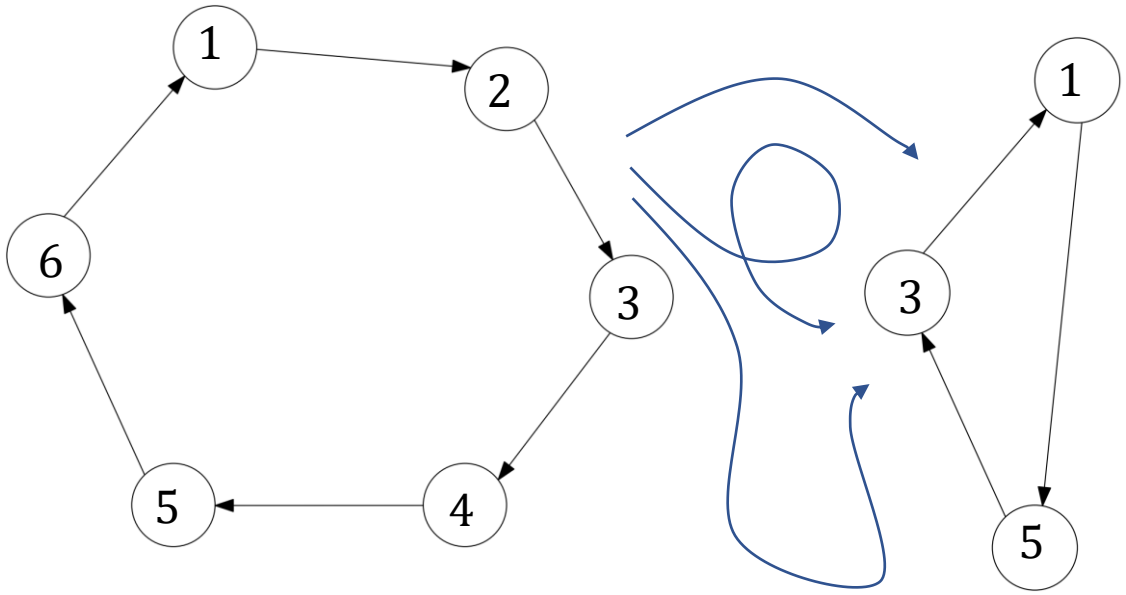


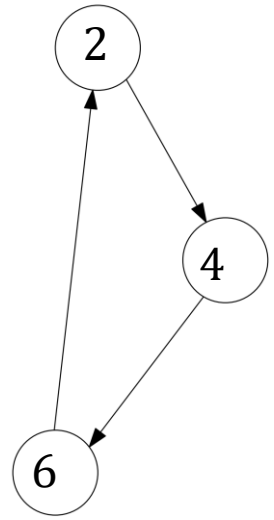
A

0	0	0	0	1	0
0	0	0	1	0	0
1	0	0	0	0	0
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	0	0

Why not just use standard NNs?

Joke credits:
Lukas Faber, John Oliver



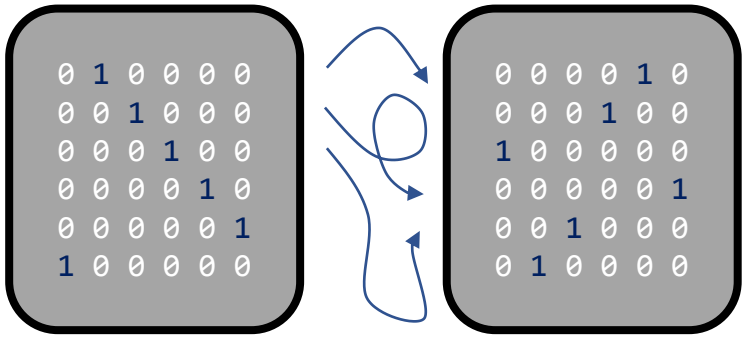


A

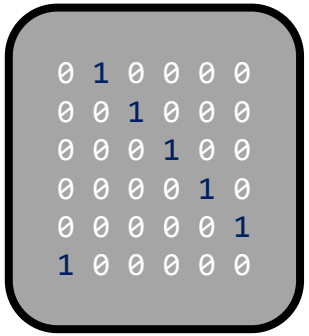
$$\begin{matrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

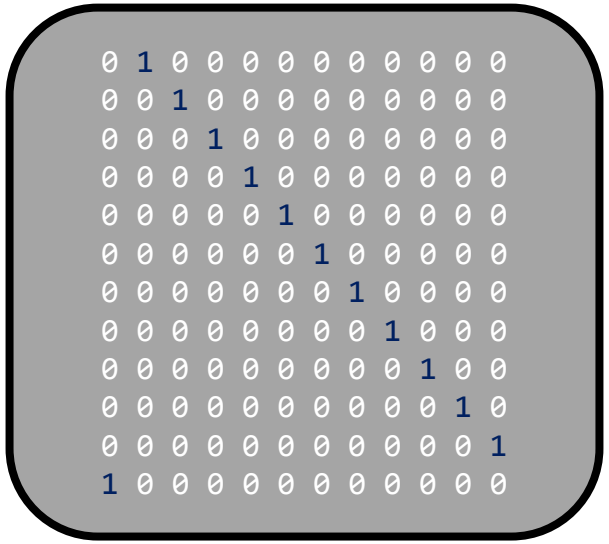
Why not just use standard NNs?

- Permutation Invariance



- Different number of nodes

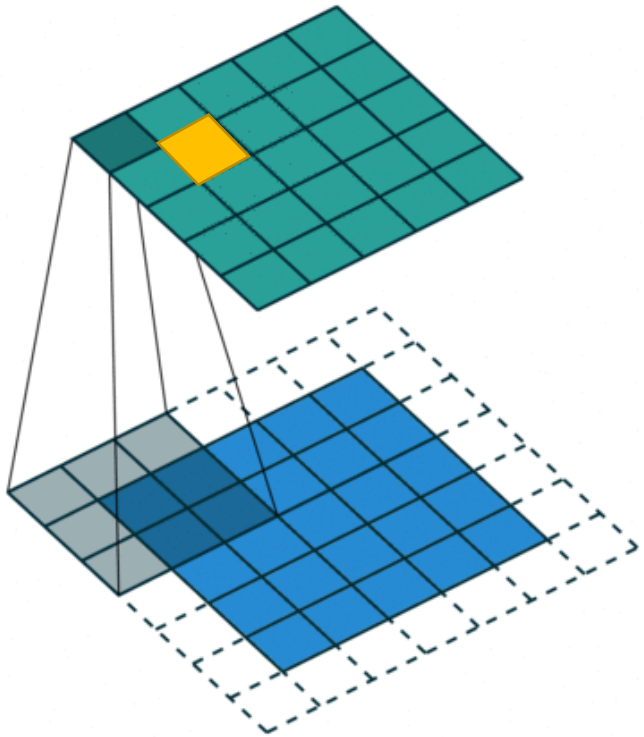


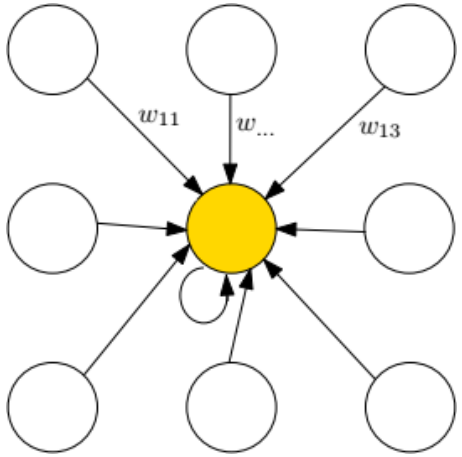


- ….

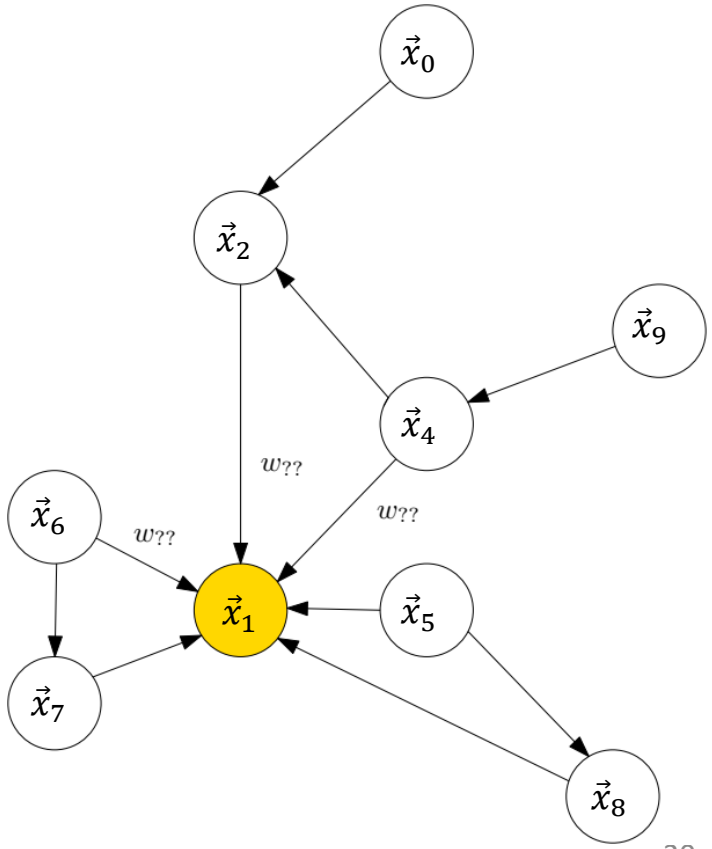
2. Graph convolution

Image convolution:



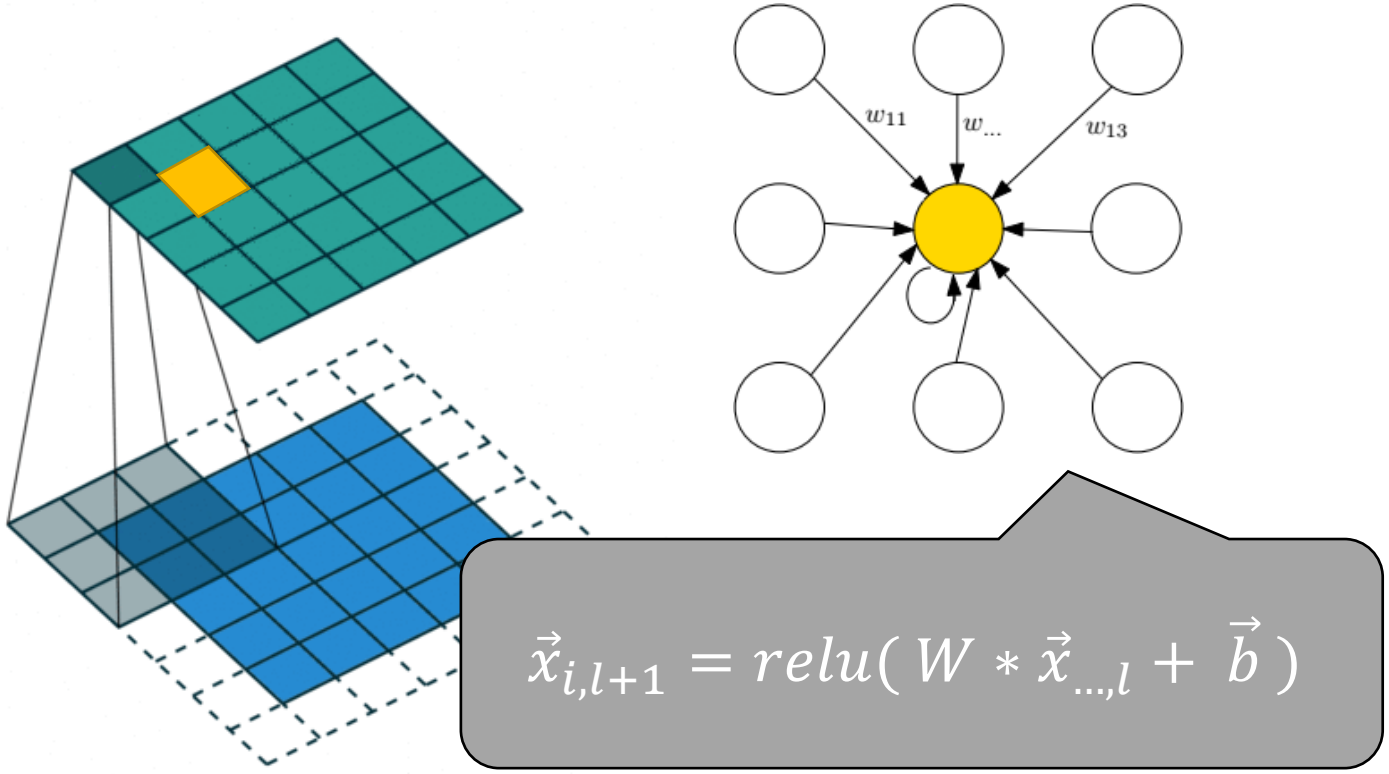


Graph convolution:

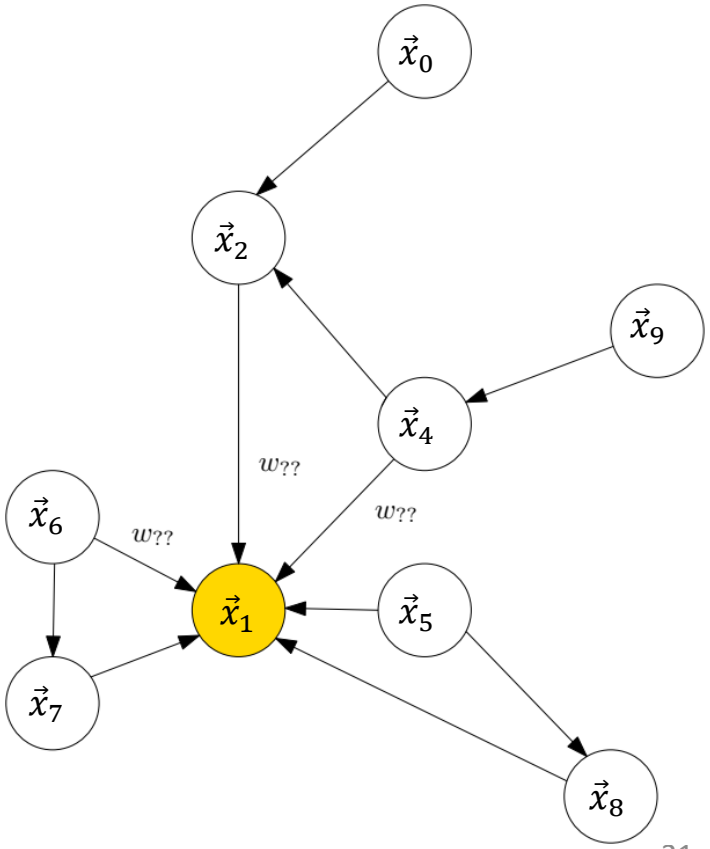


2. Graph convolution

Image convolution:



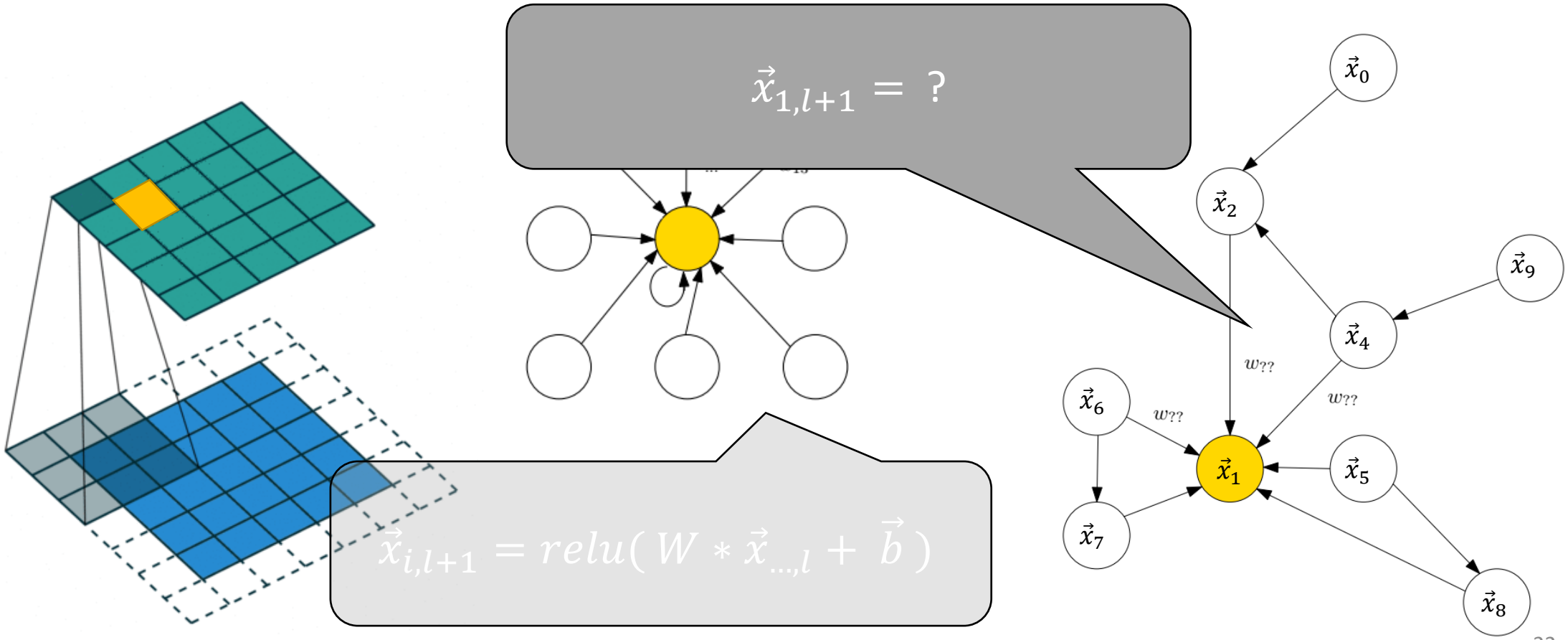
Graph convolution:



2. Graph convolution

Image convolution:

Graph convolution:

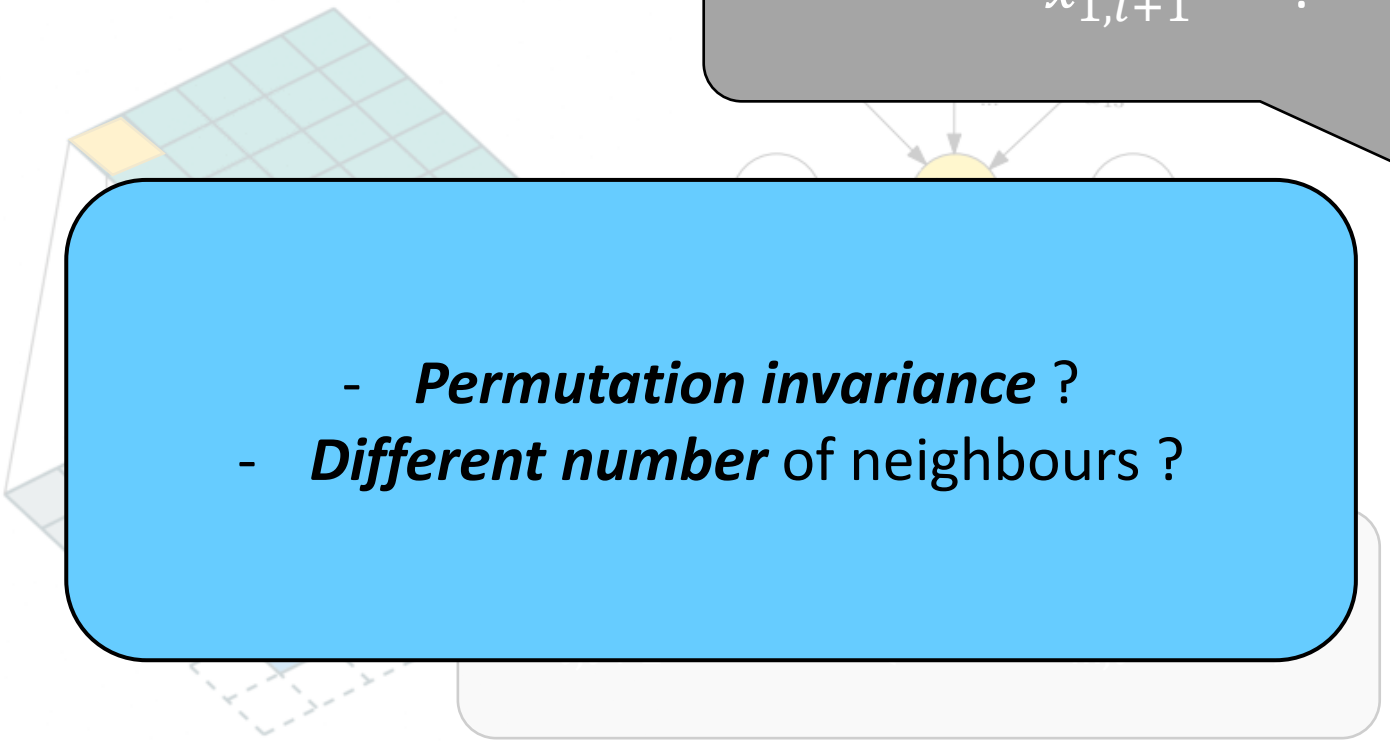


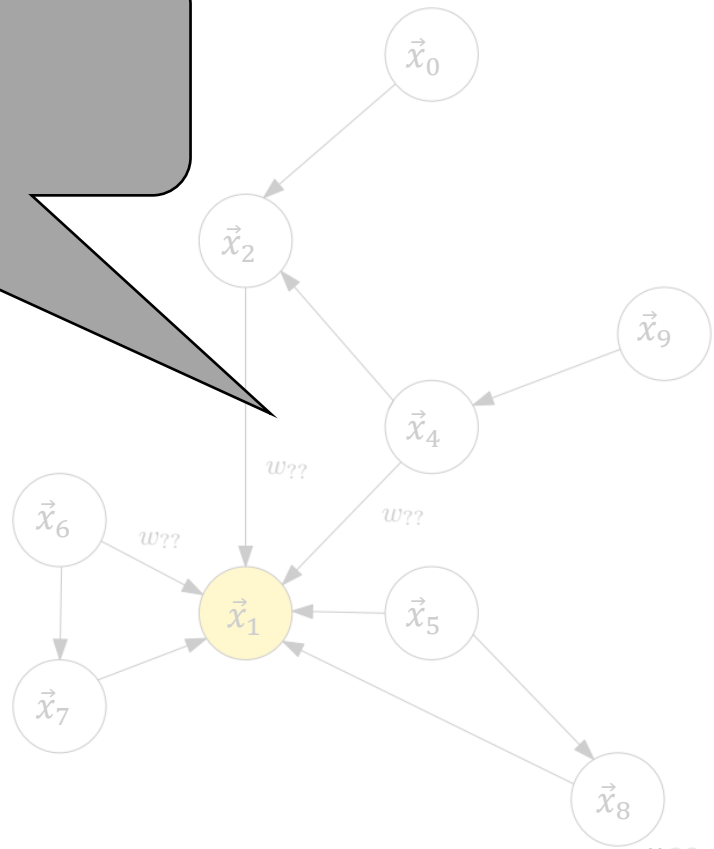
2. Graph convolution

Image convolution:

Graph convolution:

$$\vec{x}_{1,l+1} = \ ?$$

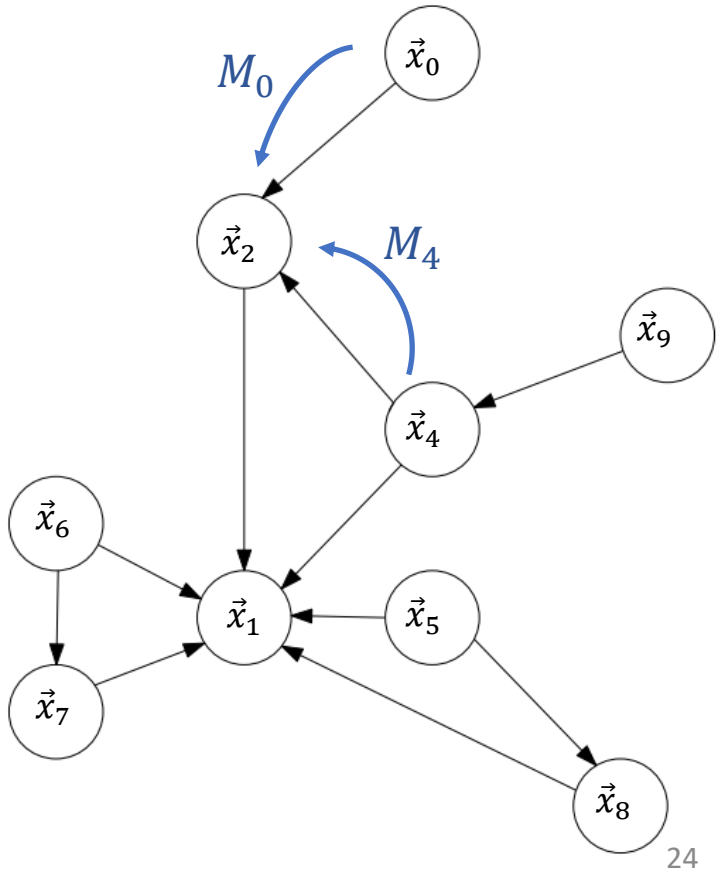




- ***Permutation invariance*** ?
- ***Different number*** of neighbours ?

Enter message passing.

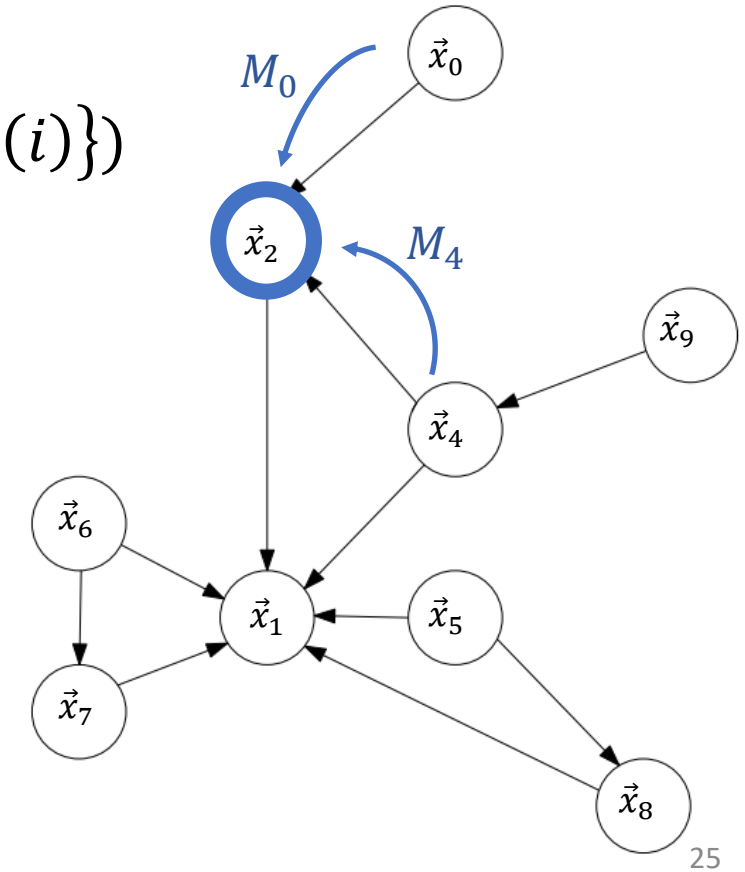
$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$



Enter message passing.

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,l} = \text{AGGREGATE}\left(\left\{M_{j,l} \mid j \in N(i)\right\}\right)$$

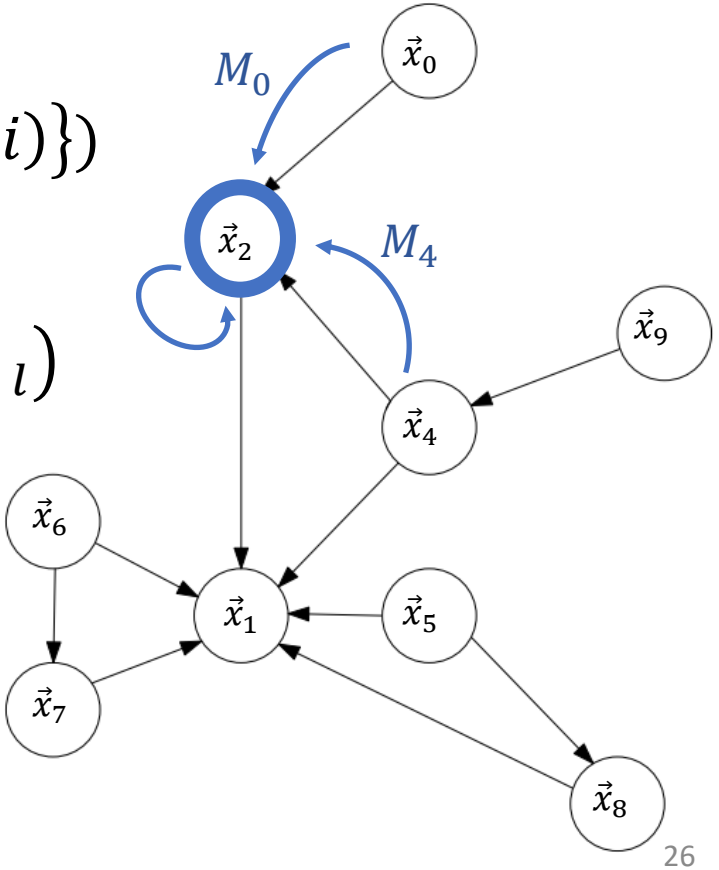


Enter message passing.

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,\,l} = \text{AGGREGATE}\left(\left\{M_{j,\,l} \,\middle|\, j \in N(i)\right\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}\left(\vec{x}_{i,\,l}\,,\ \vec{m}_{i,\,l}\right)$$



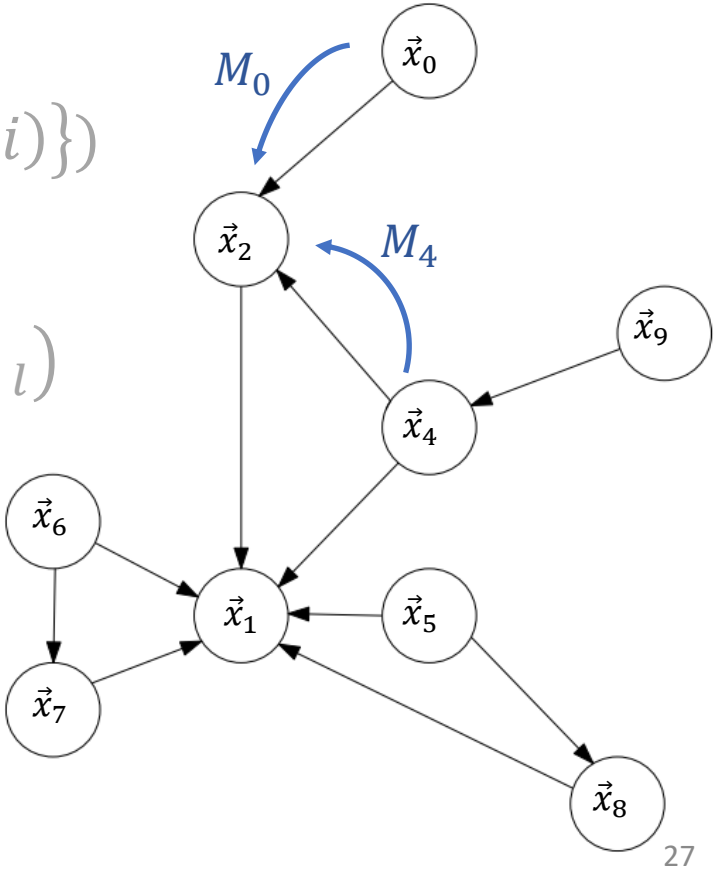
In practice:

$$M_{j,l} = \vec{x}_{j,l}$$

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,\,l} = \text{AGGREGATE}\left(\{M_{j,\,l} \mid j \in N(i)\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}\left(\vec{x}_{i,\,l},\ \vec{m}_{i,\,l}\right)$$



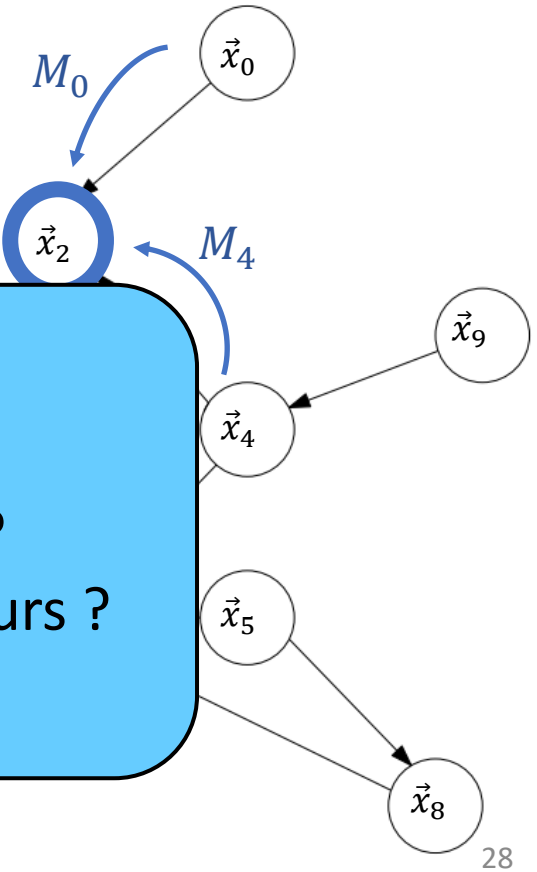
$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,l} = \text{AGGREGATE}\left(\{M_{j,l} \mid j \in N(i)\}\right)$$

How would you do it ?

Handle:

- ***Permutation invariance*** ?
- ***Different number*** of neighbours ?



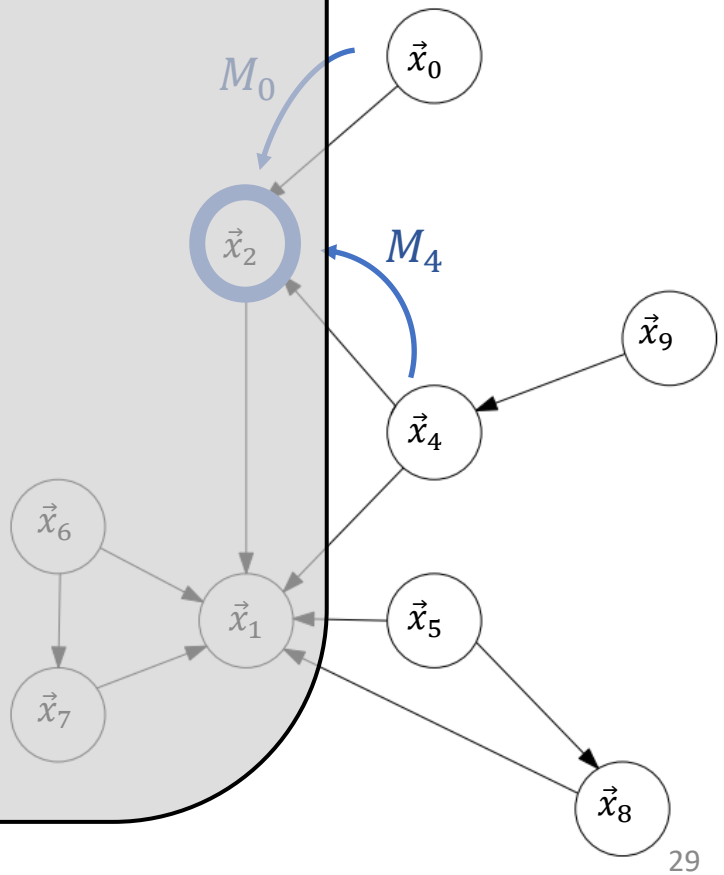
Examples

$$\vec{m}_i = \mathrm{AGGREGATE}\left(\left\{\vec{x}_j \mid j \in N(i)\right\}\right)$$

- Sum, mean, max, …
- Kipf & Welling: weighted mean:

$$\vec{m}_{j,l} = \sum_{j \in N(i)} \frac{\vec{x}_{j,l}}{\sqrt{|N(i)|\ |N(j)|}}$$

$|N(i)|$: *Number of neighbours of node i*

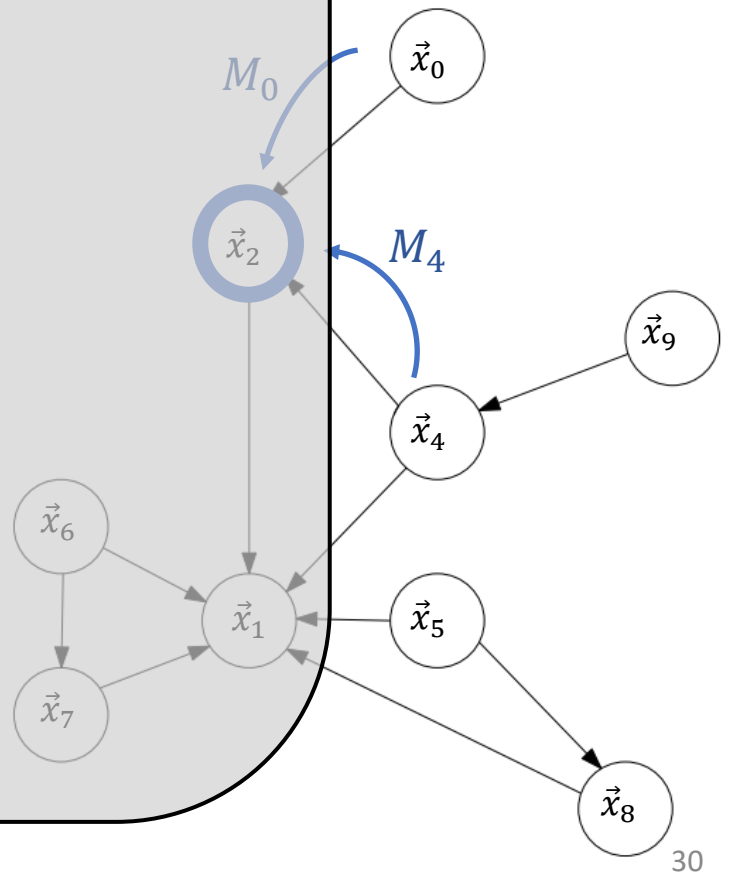


Examples

$$\vec{m}_i = \text{AGGREGATE}\left(\left\{\vec{x}_j \mid j \in N(i)\right\}\right)$$

- Sampling neighbours
- Sampling over permutations (Janossy pooling), e.g.:

$$\vec{m}_j = \sum_{\substack{some\ (j_1, j_2, \ldots j_k) \\ \in\ \boldsymbol{perm}(\boldsymbol{N}(\boldsymbol{i}))}} f(\vec{x}_{j_1}, \ldots \vec{x}_{j_k})$$



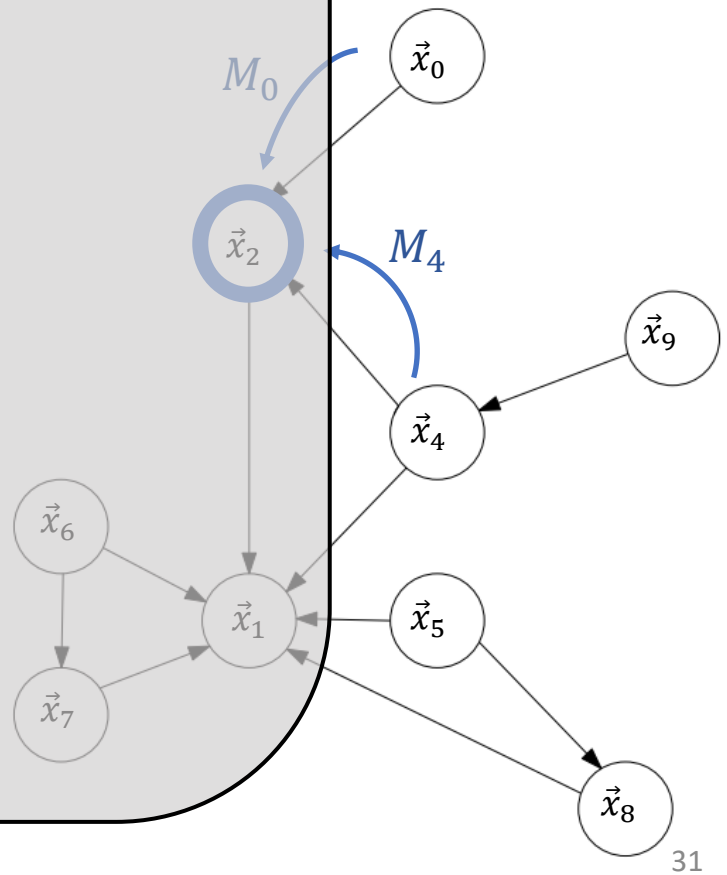
Examples

$$\vec{m}_i = \text{AGGREGATE}\left(\left\{\vec{x}_j \mid j \in N(i)\right\}\right)$$

- Theorem: f set-invariant, then: [G6]

$$f\left(\left\{\vec{x}_j \mid j \in N(i)\right\}\right) = \vec{m}_j$$

$$= \rho\left(\sum_{j \in N(i)} \varphi(\vec{x}_j)\right)$$

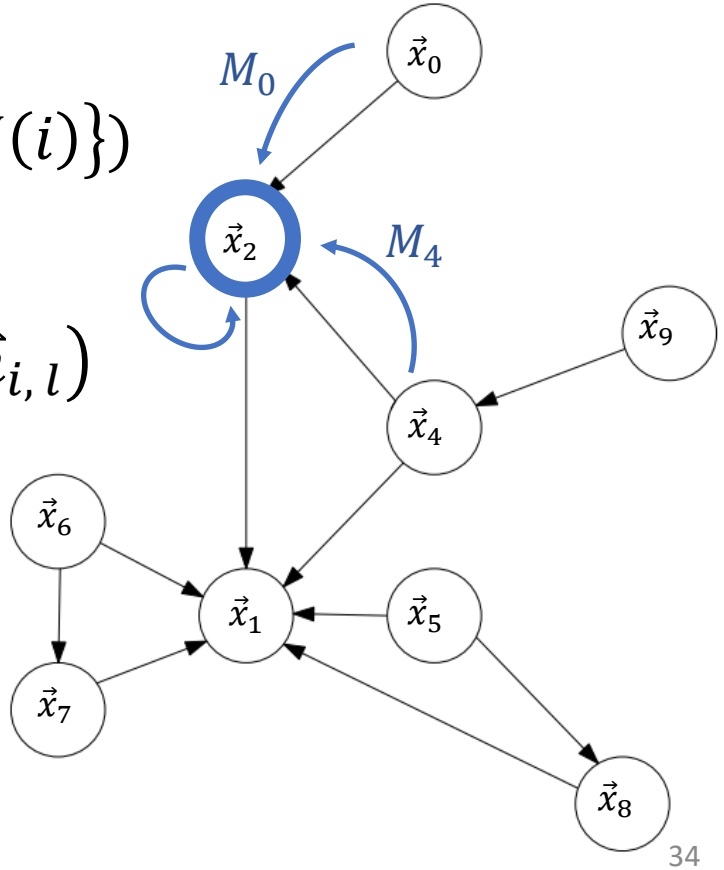


Neural network?

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,\,l} = \text{AGGREGATE}\left(\left\{M_{j,\,l} \,\middle|\, j \in N(i)\right\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}\left(\vec{x}_{i,\,l}\,,\ \vec{m}_{i,\,l}\right)$$

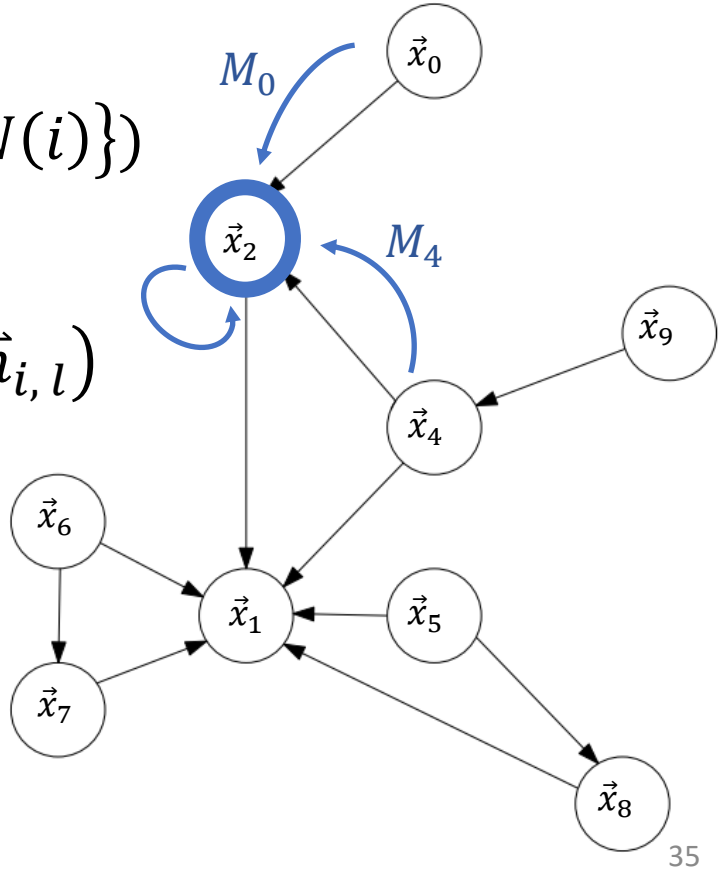


Neural network!

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,l} = \text{AGGREGATE}_{\varphi}(\{M_{j,l} \mid j \in N(i)\})$$

$$\vec{x}_{i,l+1} = \text{UPDATE}_{\theta}(\vec{x}_{i,l}, \vec{m}_{i,l})$$

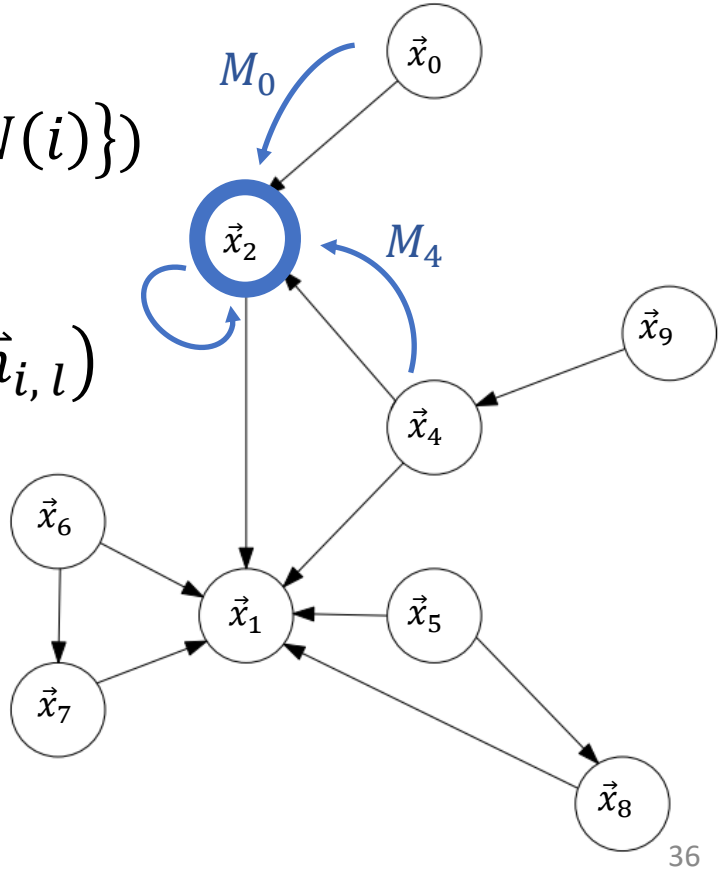


L iterations

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,\,l} = \text{AGGREGATE}_{\varphi}\left(\left\{M_{j,\,l} \,\middle|\, j \in N(i)\right\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}_{\theta}\left(\vec{x}_{i,\,l}\,,\ \vec{m}_{i,\,l}\right)$$



$\vec{x}_{i,\,L}$

...

3. GNNs: Extensions

Attention: GAT [G1, 2]

Skip connections [G1, 3, 4]

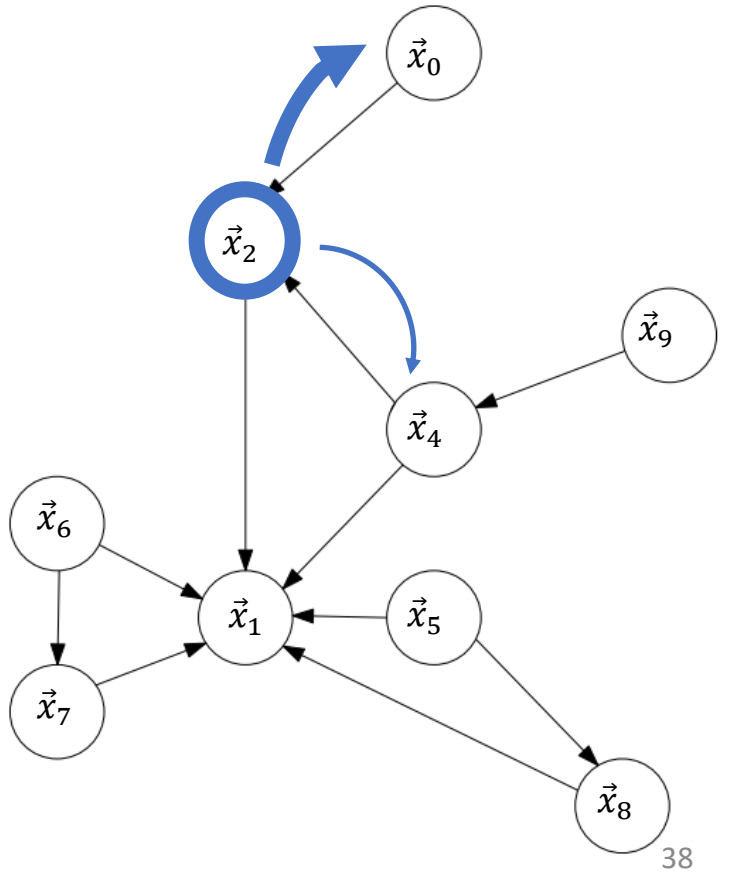
Gated GNNs [G1, 8]

Relational GNNs [G1, 5, 6, 7]

GAT: Graph Attention Networks [G1, 2]

Why?

- Not all neighbours are the same!

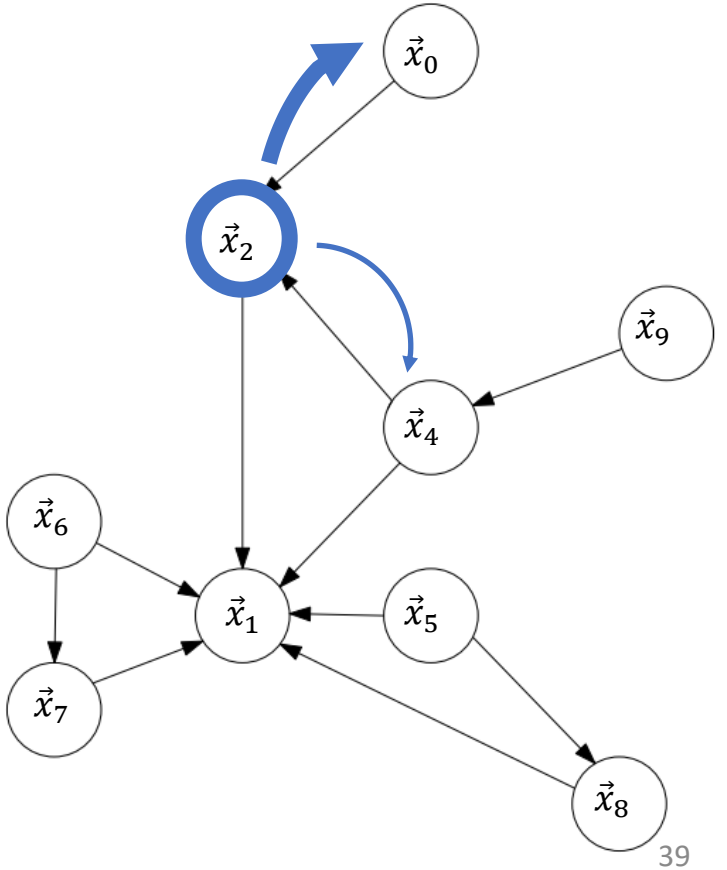


GAT: Graph Attention Networks

[G1, 2]

Why?

- Not all neighbours are the same!



How would you do it?

GAT: Graph Attention Networks

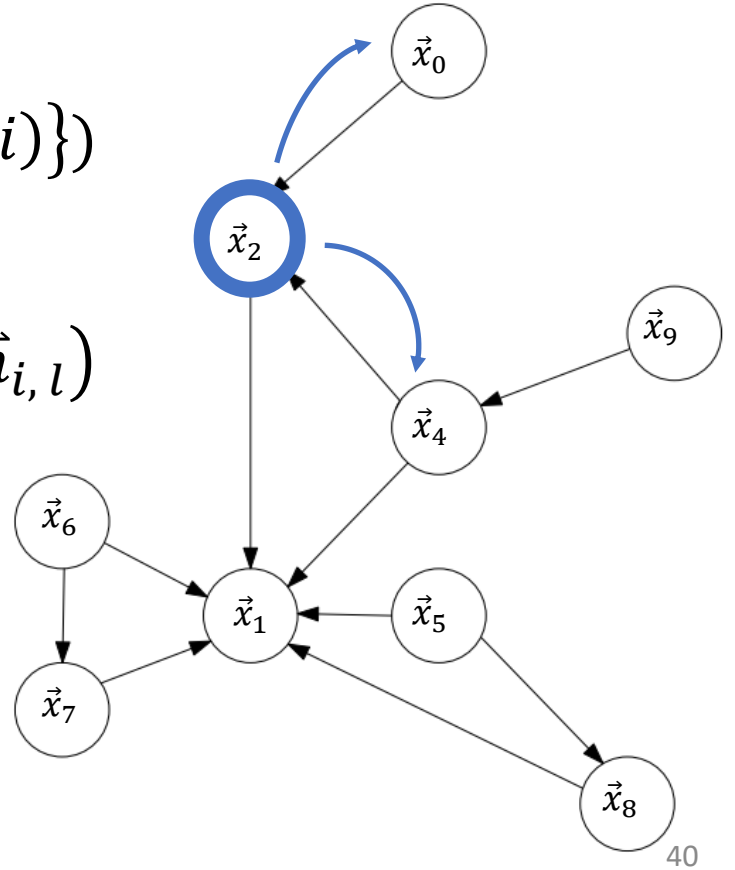
[G1, 2]

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,\,l} = \text{AGGREGATE}_{\varphi}\left(\left\{\, \vec{\text{x}}_{\text{j}} \;\middle|\; j \in N(i) \right\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}_{\theta}\left(\vec{x}_{i,\,l}\,,\ \vec{m}_{i,\,l}\right)$$

How would you do it?



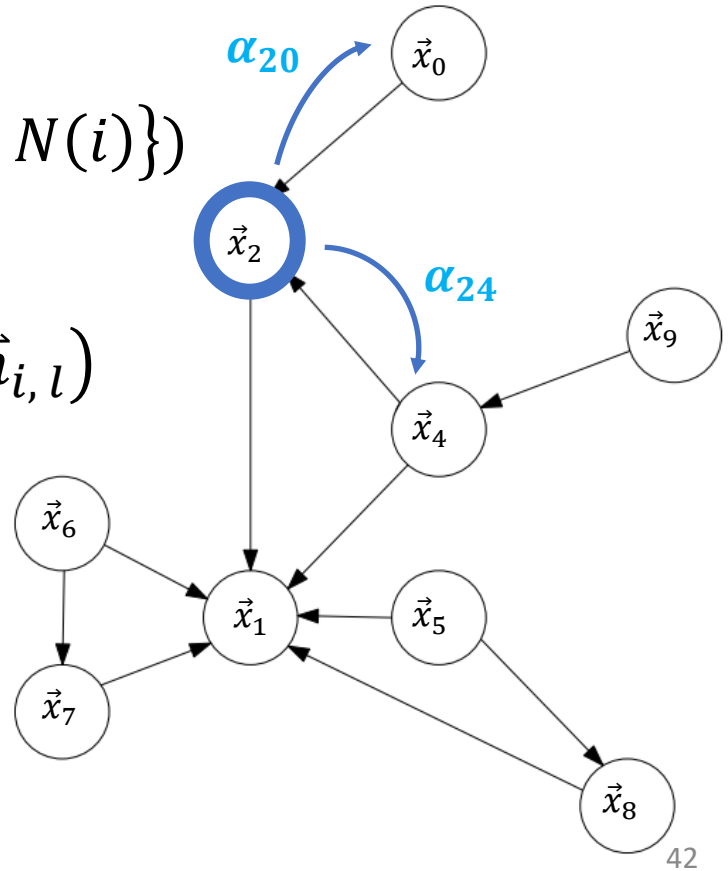
GAT: Graph Attention Networks

[G1, 2]

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

$$\vec{m}_{i,\,l} = \text{AGGREGATE}_{\varphi}\left(\left\{\boldsymbol{\alpha_{i,j}}\ \vec{\text{x}}_{\text{j}} \;\middle|\; j \in N(i)\right\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}_{\theta}\left(\vec{x}_{i,\,l}\,,\ \vec{m}_{i,\,l}\right)$$



$\boldsymbol{\alpha_{i,j}}$: Attention weights,

e.g. via softmax, transformer-like, …

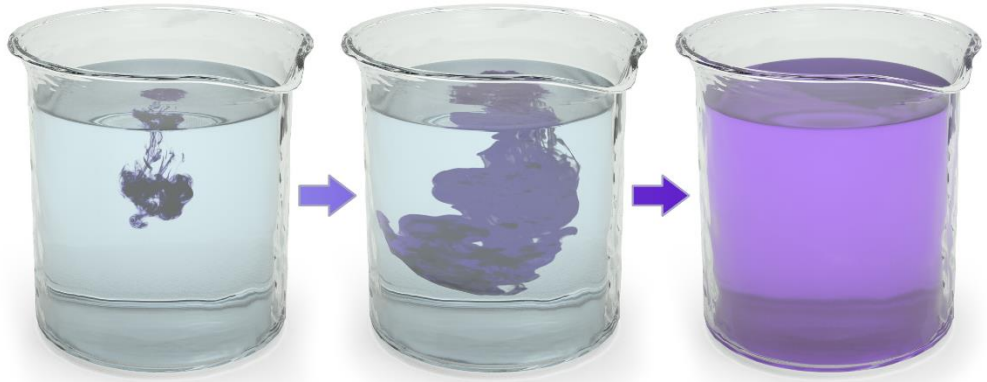
Skip connections (residual)

[G1, 3, 4]

Why?

- Oversmoothing
- Gradient stability

  (multi-layer GNNs are RNNs)



Skip connections (residual)

[G1, 3, 4]

Why?

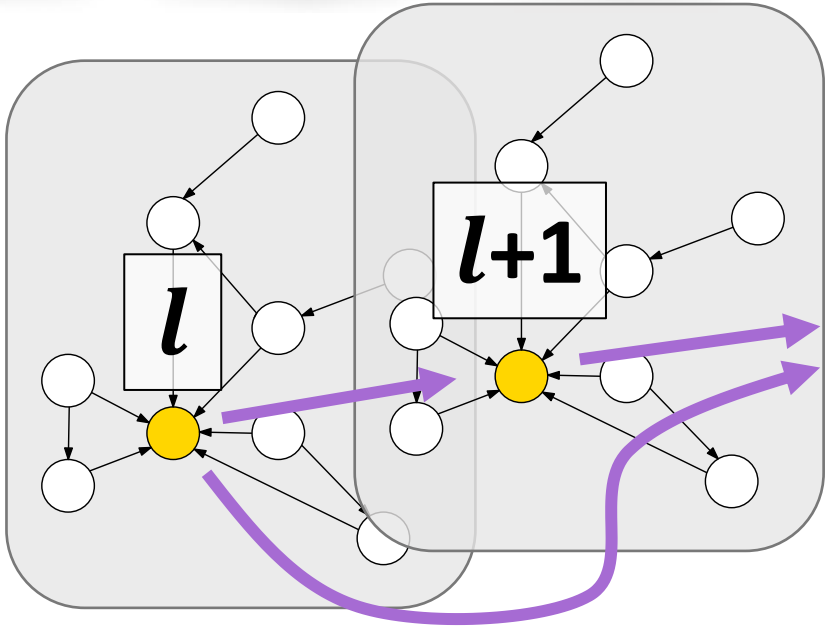
- Oversmoothing
- Gradient stability

  (multi-layer GNNs are RNNs)

How:

Stack with previous

Final embedding = stack of features



Gated GNNs

[G1, 8]

$$M_{j,l} = \text{MESSAGE}(\vec{x}_{j,l})$$

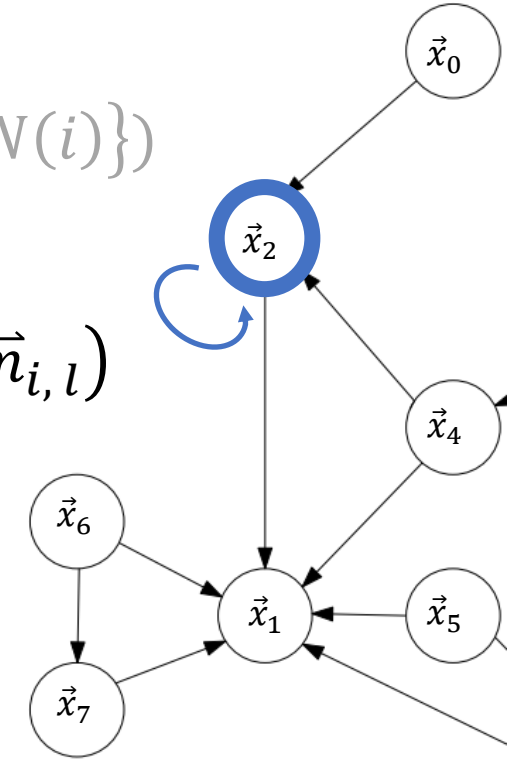
Why?

- Oversmoothing
- Gradient stability

  (multi-layer GNNs are RNNs)

$$\vec{m}_{i,\,l} = \text{AGGREGATE}_{\varphi}\left(\{M_{j,\,l} \mid j \in N(i)\}\right)$$

$$\vec{x}_{i,\,l+1} = \text{UPDATE}_{\theta}\left(\vec{x}_{i,\,l}\,,\ \vec{m}_{i,\,l}\right)$$



How:

$\text{UPDATE} \rightarrow$ RNN unit:

$$\vec{x}_{i,\,l} = \boldsymbol{GRU}(\vec{x}_{i,\,l-1},\ \vec{m}_{i,\,l}))$$

$$\vec{x}_{i,\,l} = \boldsymbol{LSTM}(\vec{x}_{i,\,l-1},\ \vec{m}_{i,\,l})$$

Relational (G)NNs [G1, 5, 6, 7]

Two meanings:

- **Different relation type ⟷ different weights** [G1, 6]
- **« Edge embeddings » as output** [5, 6, 7]

Relational (G)NNs [G1, 5, 6, 7]

Two meanings:

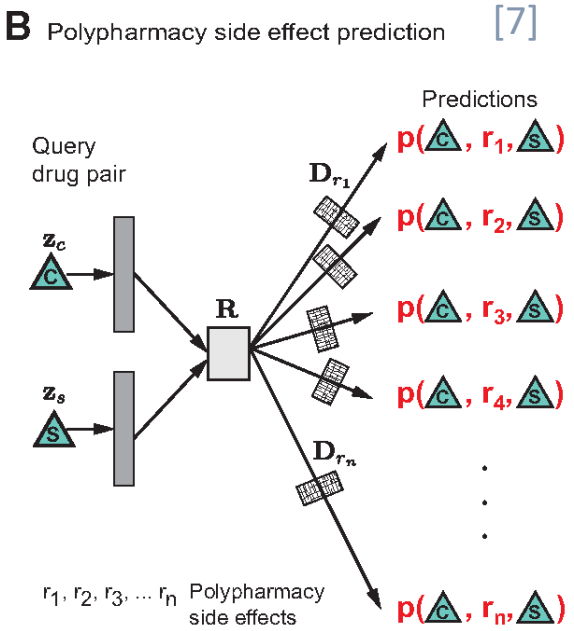
- **Different relation type ⟷ different weights** [G1, 6]
- **« Edge embeddings » as output** [5, 6, 7]

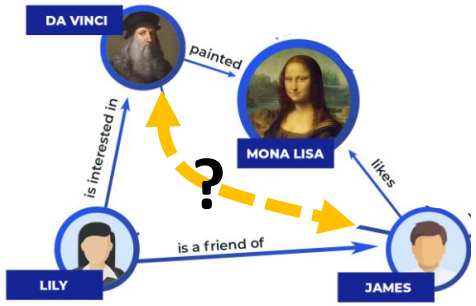
Relational (G)NNs

[G1, 5, 6, 7]

Two meanings:

- Different relation type ⟷ different weights
- **« Edge embeddings » as output** [5, 6, 7]





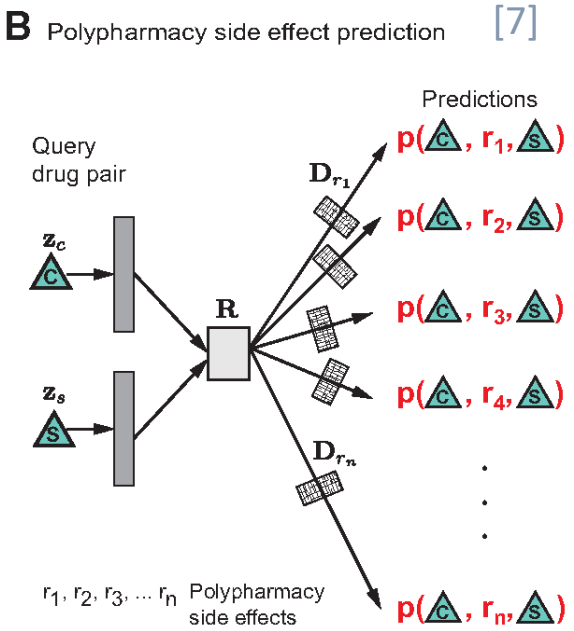
[~6]

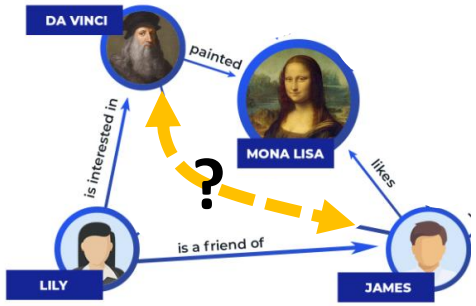
Relational (G)NNs

[G1, 5, 6, 7]

Two meanings:

- Different relation type ⟷ different weights
- **« Edge embeddings » as output** [5, 6, 7]
  - Not a graph but a building block





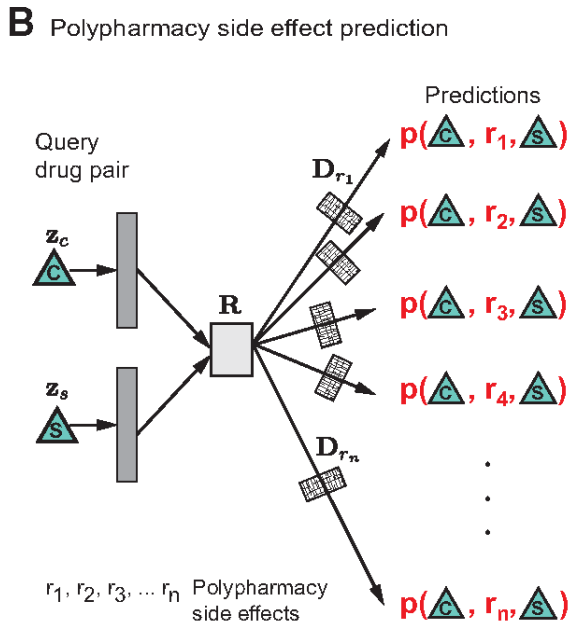
[~6]

Relational (G)NNs

[G1, 5, 6, 7]

Two meanings:

- Different relation type ⟷ different weights
- **« Edge embeddings » as output** [5, 6, 7]



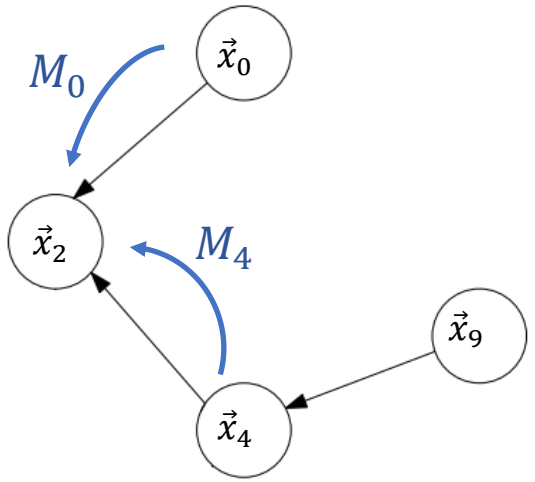
r_{ij} : **Edge label**

- ***e.g.*** $out(i,j) = \vec{x}_i W_{r_{ij}} \vec{x}_j$

$out(i,j) = MLP_{\varphi}(\vec{x}_i, \vec{x}_j)$

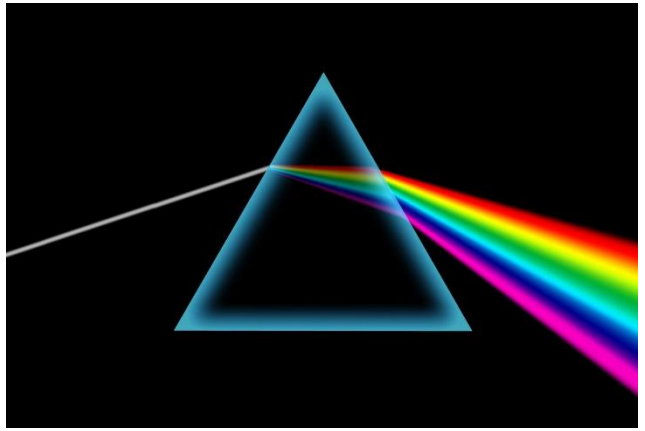
You just saw:

‘Localized’ graph convolution

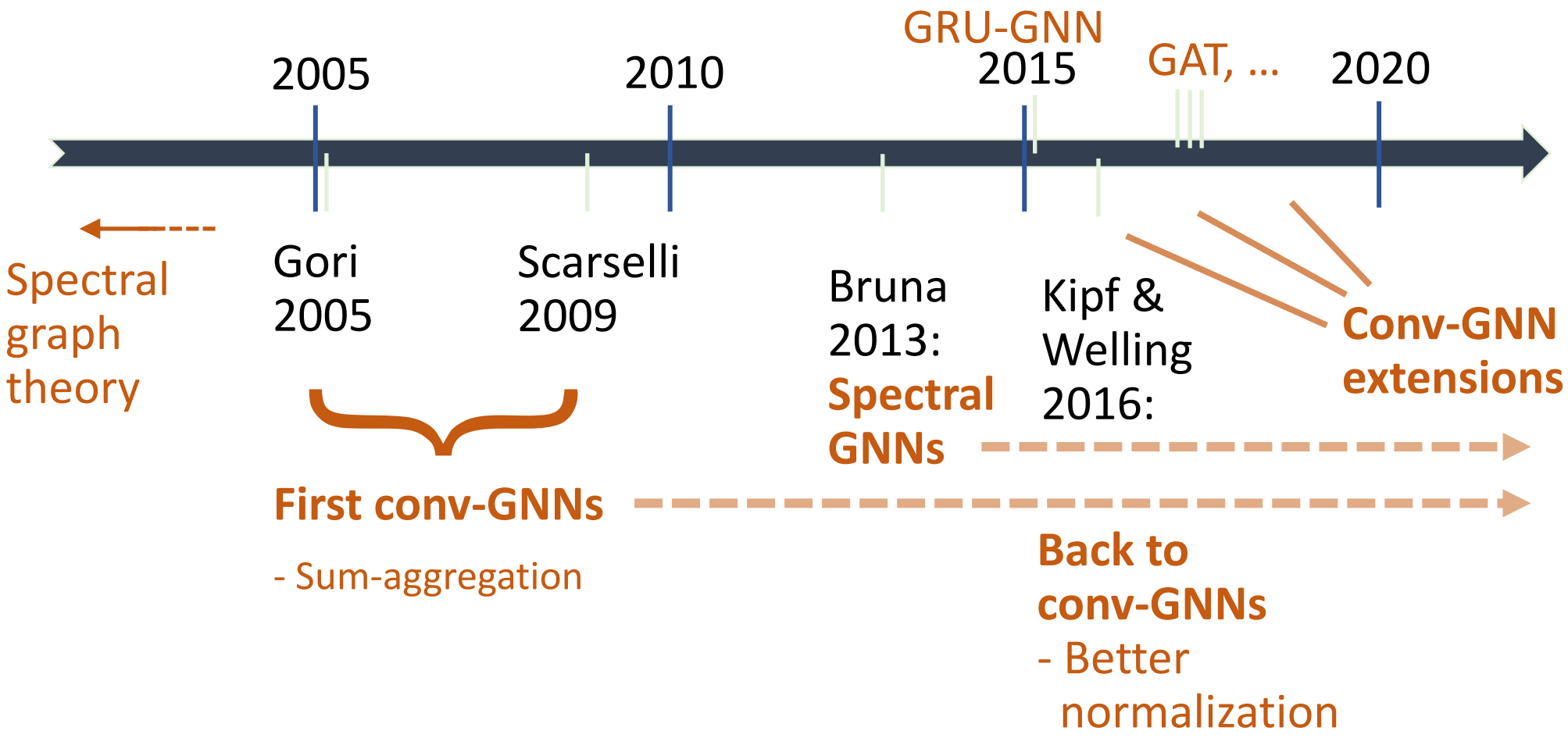


There is also:

Spectral graph convolution

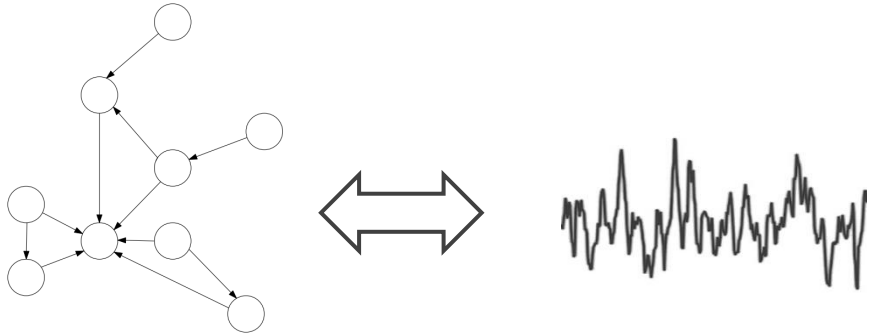


GNNs: A short history



4. Spectral Graph Convolutional Networks

Idea: Analogy to Fourier Analysis



- Convolution ⟺ multiplication in the frequency domain
- 'Frequency domain': Change of basis, $\hat{x} = U^T \vec{x}$; $\vec{x} = U U^T \vec{x}$

  → « Convolution » on a graph:

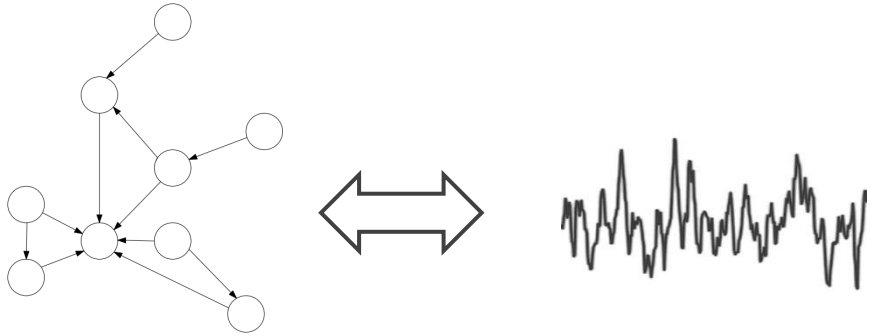
$$\vec{x}_{l+1} = U \, diag(\vec{\theta}) \, U^T \vec{x}_l$$

$$U = [\vec{u}_1, \vec{u}_2, \ldots \vec{u}_N]$$

(*basis vectors*)

4. Spectral Graph Convolutional Networks

Idea: Analogy to Fourier Analysis



- Convolution $\Leftrightarrow$ multiplication in the frequency domain
- 'Frequency domain': Change of basis, $\hat{x} = U^T \vec{x}; \quad \vec{x} = U U^T \vec{x}$

→ « Convolution » on a graph:

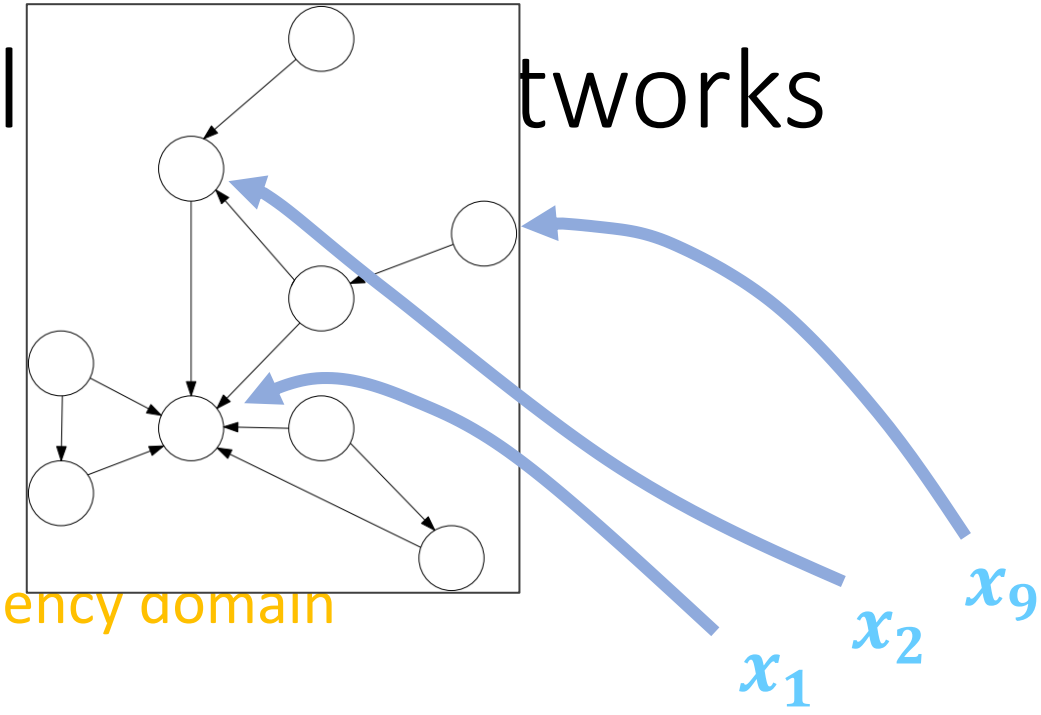
$$\vec{x}_{l+1} = U\, diag(\vec{\theta})\, U^T \vec{x}_l$$

Scalar multiplication

$$\vec{u}_i \rightarrow \theta_i \vec{u}_i$$

i-th "frequency"

4. Spectral Graph Convol tworks



Idea: Analogy to Fourier Analysis

- Convolution ⟺ multiplication in the frequency domain
- 'Frequency domain': Change of basis, $\hat{x} = U^T\vec{x}$; $\vec{x} = UU^T\vec{x}$

→ « Convolution » on a graph:

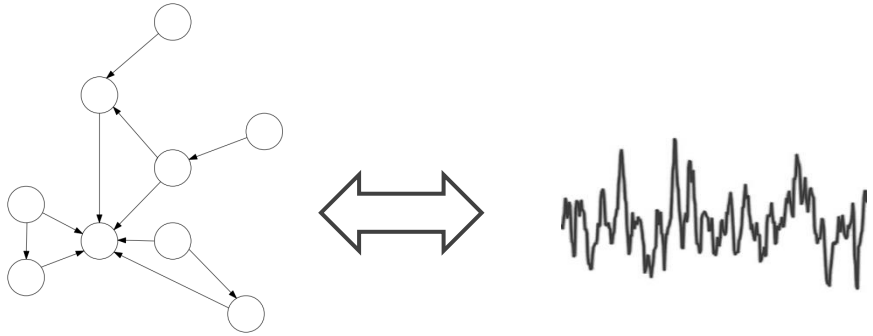
$$\vec{u}_i \rightarrow \theta_i\vec{u}_i$$

$$\vec{x}_{l+1} = U\, diag(\vec{\theta})\, U^T\vec{x}_l$$

$$\vec{x}: \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_9 \end{bmatrix}$$

4. Spectral Graph Convolutional Networks

Idea: Analogy to Fourier Analysis



- Convolution $\Leftrightarrow$ multiplication in the frequency domain
- 'Frequency domain': Change of basis, $\hat{x} = U^T \vec{x}; \quad \vec{x} = UU^T \vec{x}$

  → « Convolution » on a graph:

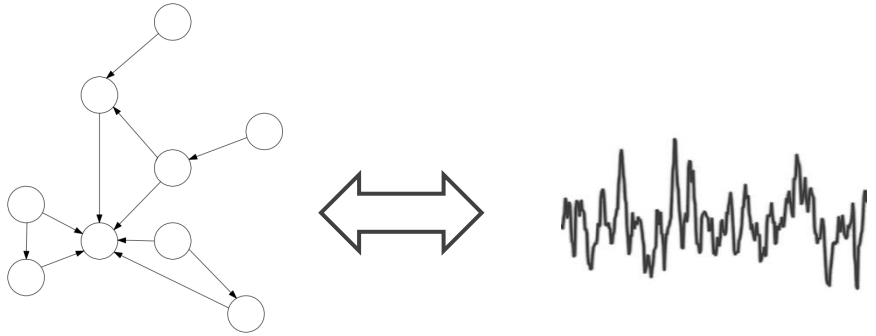
$$\vec{u}_i \rightarrow \theta_i \vec{u}_i$$

$$\vec{x}_{l+1} = \underbrace{U\, diag(\vec{\theta})\, U^T \vec{x}_l}$$

Mixes all nodes

4. Spectral Graph Convolutional Networks

Localizing spectral graph convolution:



ency domain

- 'Frequency domain': Change of basis, $\hat{x} = U^T \vec{x}$; $\vec{x} = U U^T \vec{x}$

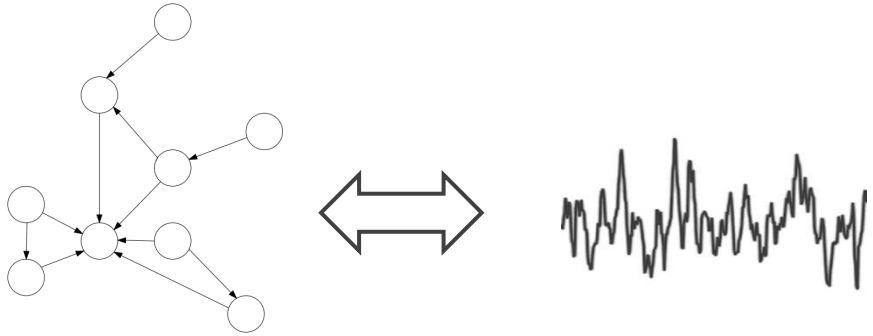
→ « Convolution » on a graph:

$$\vec{u}_i \to \theta_i \vec{u}_i$$

$$\vec{x}_{l+1} = U \, diag(\vec{\theta}) \, U^T \vec{x}_l$$

4. Spectral Graph Convolutional Networks

Localizing spectral graph convolution:



ency domain

- ’Frequency domain’: Change of basis, $\hat{x} = U^T \vec{x}$; $\vec{x} = UU^T \vec{x}$

→ « Convolution » on a graph:

$$\vec{u}_i \rightarrow \theta_i \vec{u}_i$$

$$\vec{x}_{l+1} = \boldsymbol{L^k}\ \vec{x}_l$$

(simplified)

$\boldsymbol{L}$: Graph Laplacian,
combines 1-hop neighbours

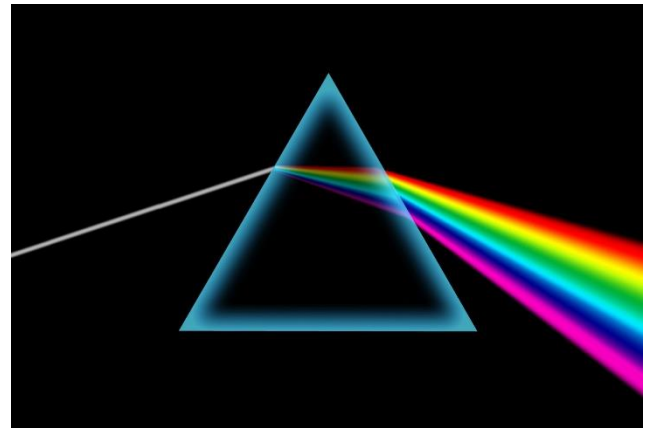
4. Spectral Graph Convolutional Networks

Pros and Cons:

(+) Is not local

(+) Can be localized

(+) Weights $\vec{\theta}$ can be transferred to other graphs *



(-) * Sometimes

5. What does not yet work?

- $\mathrm{AGGREGATE}$: Retain information about number of neighbours?

Kipf & Welling:

$$\vec{\mathrm{m}}_{\mathrm{j},\,l} = \sum_{\mathrm{j} \in N(\mathrm{i})} \frac{\vec{\mathrm{x}}_{\mathrm{j},\,l}}{\sqrt{|N(i)|\ |N(j)|}}$$

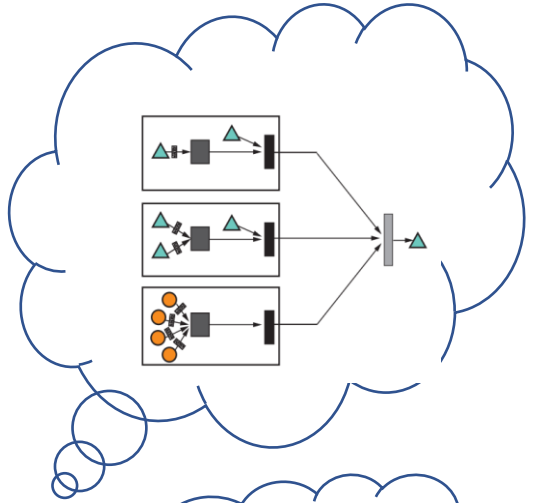
$|N(i)|$ ***: Number of neighbours of node i***

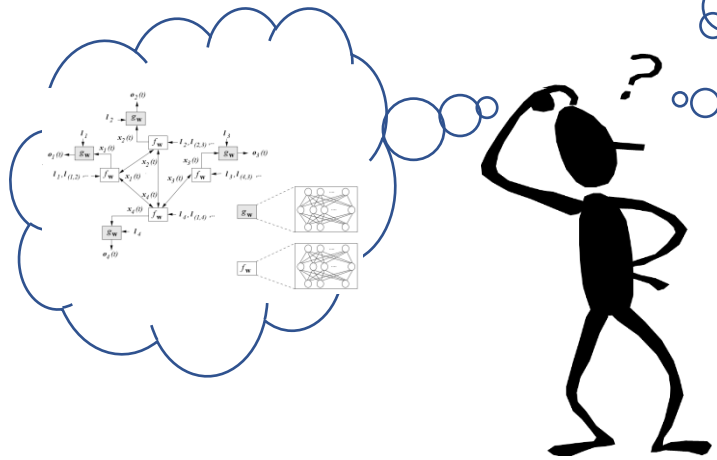
5. What does not yet work?

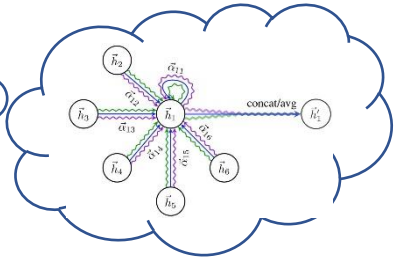
- $\mathrm{AGGREGATE}$: Retain information about number of neighbours?
- Shallowness - typical GNNs: **2**-10 layers
  - Limited visual field
- Spectral methods: Interpretation?

5. What does not yet work?

- GNNs cannot express everything (see next week)
- Oversmoothing (see next week)
- « Best » model not yet determined
- Parallelizing graph convolutions



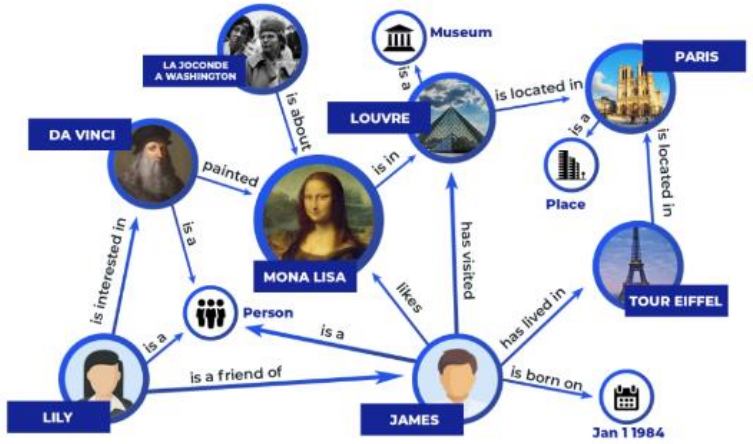




6. Conclusion: What are GNNs good for?

- As computation module, aligned with graph-structured tasks

- If you have graph data



https://medium.com/neuralspace/graphs-neural-networks-in-nlp-dc475eb089de

https://favpng.com/png_view/calculator-calculator-clip-art-png/QTsQeybd

7. Discussion starters

- (How) Would you integrate the number of neighbours into the network?
- Spectral GCNs or conv-GNNs? ☺
- How to know whether we need to integrate many-hop neighbours?
- Language models: To integrate (syntax-graph) structure, or not?
- Generally, do you think it helps to add structure to a problem? What are advantages, what are disadvantages?

7. Discussion starters

- What architecture combination would you want to try? For what?
- (How) Could you use a GNN for a graph-structured problem where you do not know the graph?
- How important do you think are GNNs? Will they be the next big thing, or irrelevant? Why?
- What would be other approaches to graph neural networks, that you can think of? Are there any?

Thank you.

Waiting for your questions
on Tue at 11:30!

More image credits

3rd slide:

- https://blog.equinix.com/blog/2017/05/08/how-to-segment-traffic-flows-at-the-network-edge/
- https://arxiv.org/pdf/1703.04826.pdf
- https://www.researchgate.net/figure/CK1-in-circadian-rhythm-regulation-By-binding-of-the-BMAL1-CLOCK-heterodimer-to-the_fig1_262931114
- https://medium.com/neuralspace/graphs-neural-networks-in-nlp-dc475eb089de
- https://en.wikipedia.org/wiki/Protein_structure#/media/File:Domain_Homology.png
- https://georgemartinapprentice.wordpress.com/author/georgemartinapprentice/page/2/

More image credits

Remainder:

- https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d
- https://makeagif.com/gif/magmaheatdiffusion-8tuaGk?position=1
- https://medium.com/neuralspace/graphs-neural-networks-in-nlp-dc475eb089de
- https://www.ni.com/de-ch/innovations/white-papers/06/find-the-right-analog-signal-generator.html
- https://www.dkfindout.com/us/science/light/splitting-light/
- http://www.porterco.org/images/pages/N173/Stick%20Figure%20Question%20Mark.png
- https://mila.quebec/en/publication/graph-attention-networks/30207/

References

"Standard" CGN:
[1] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *ICLR 2017, arXiv:1609.02907*.

GAT:
[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Skip connections:
[3] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*.

[4] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. I., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning* (pp. 5453-5462). PMLR.

Relational (G)NNs:
[5] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., & Lillicrap, T. (2017). A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*.

[6] Schlichtkrull. In *European semantic web conference* (pp. 593-607). Springer, Cham. , M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018, June). Modeling relational data with graph convolutional networks

[7] Zitnik, M., Agrawal, M., & Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics, 34*(13), i457-i466.

References

Gated GNNs:

[8] Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint* *arXiv:1511.05493*.

[9] Marcheggiani, D., & Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint* *arXiv:1703.04826*.

References

Data sets:

[D1] Benchmark graph datasets: https://chrsmrrs.github.io/datasets/

[D2] List of citation-graph-like datasets: https://github.com/shchur/gnn-benchmark#datasets

[D3] Debnath et al. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, *34*(2), 786-797. (<-- MUTAG Dataset)

[D4] Kazius, J., McGuire, R., & Bursi, R. (2005). Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, *48*(1), 312-320. Online: https://pubs.acs.org/doi/abs/10.1021/jm040835a

References: General

[G1] Ch. “The Graph Neural Network Model”, *W. Hamilton, Graph Representation Learning [Draft]*
https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_mar29.pdf
https://www.cs.mcgill.ca/~wlh/grl_book/

Reviews:

[G2] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., ... & Sun, M. (2018). Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

[G3] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*. *arXiv:1901.00596*

For comparison of some AGGREGATE functions:
[G4] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. *arXiv preprint arXiv:1810.00826*.

First GNN
[G5] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks, 20*(1), 61-80.

Thm permutation invariant functions:
[G6] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., & Smola, A. (2017). Deep sets. *arXiv preprint arXiv:1703.06114*.

References: Spectral graph convolution

[S1] Hammond, D. K., Vandergheynst, P., & Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, *30*(2), 129-150.

[S2] Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips* (Vol. 14, No. 14, pp. 585-591).

[S3] Shuman, D. I., Ricaud, B., & Vandergheynst, P. (2012). A windowed graph Fourier transform. In *2012 IEEE Statistical Signal Processing Workshop (SSP)* (pp. 133-136).

[S4] Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Transfer across graphs:

[S5] Bianchi, F. M., Grattarola, D., Livi, L., & Alippi, C. (2021). Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

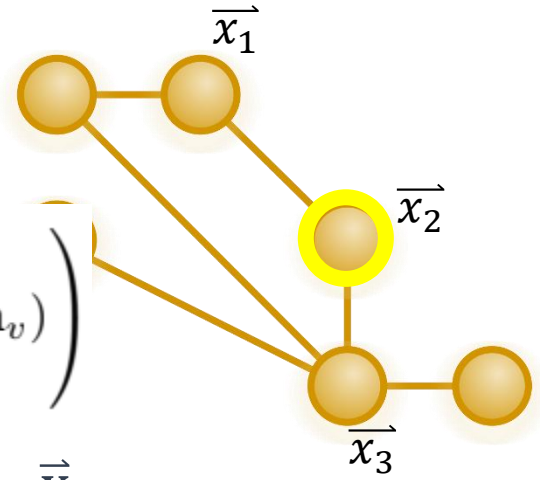
Appendix

How would *you* do it?

$$\vec{m}_{j,t} = \text{AGGREGATE}\left(\{\vec{x}_{j,t} \mid j \in N(i)\}\right)$$

$$\vec{x}_{i,t+1} = \text{UPDATE}\left(\vec{x}_{j,t},\ \vec{m}_{j,t}\right)$$

N(i): Neighbours of i



AGGREGATE() examples:

$$\mathbf{m}_{\mathcal{N}(u)} = \text{MLP}_\theta\left(\sum_{v \in N(u)} \text{MLP}_\phi(\mathbf{h}_v)\right)$$

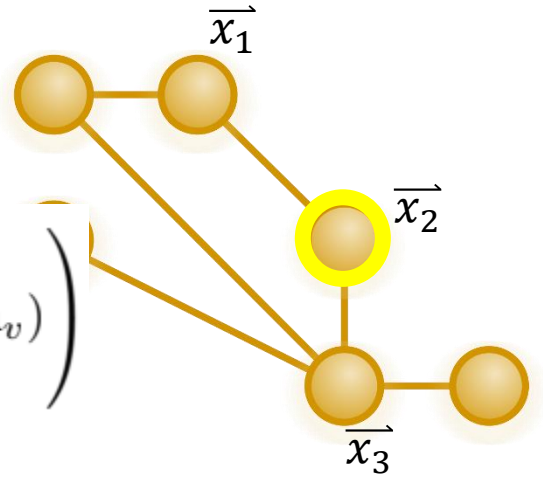
- Sum, mean, max
- Kipf & Welling: Weighted mean $\vec{m}_{j,t} = \sum_{j \in N(i)} \frac{\vec{x}_{j,t}}{\sqrt{|N(i)|\ |N(j)|}}$
- Sampling neighbours
- Sampling over permutations (Janossy pooling)

How would *you* do it?

$$\vec{m}_{j,t} = \text{AGGREGATE}\left(\{\vec{x}_{j,t} \mid j \in N(i)\}\right)$$

$$\vec{x}_{i,t+1} = \text{UPDATE}\left(\vec{x}_{j,t},\ \vec{m}_{j,t}\right)$$

N(i): Neighbours of i



AGGREGATE() examples:

$$\mathbf{m}_{\mathcal{N}(u)} = \text{MLP}_\theta\left(\sum_{v \in N(u)} \text{MLP}_\phi(\mathbf{h}_v)\right)$$

Complete iteration, one example:

(different notation: x --> h)

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right)$$

GAT: Graph Attention Networks [2]

Why?

How:

$$\vec{\mathrm{m}}_{\mathrm{j}} = \sum_{\mathrm{j} \in \mathrm{N(i)}} \alpha_{i,j} \vec{\mathrm{x}}_{\mathrm{j}}$$

Remember:

$$\vec{\mathrm{m}}_{\mathrm{j,\,t}} = \mathrm{AGGREGATE}\left(\left\{\vec{\mathrm{x}}_{\mathrm{j,\,t}} \mid \mathrm{j} \in \mathrm{N(i)}\right\}\right)$$

$$\vec{\mathrm{x}}_{\mathrm{i,\,t+1}} = \mathrm{UPDATE}\left(\vec{\mathrm{x}}_{\mathrm{j,\,t}}, \vec{\mathrm{m}}_{\mathrm{j,\,t}}\right)$$

$\alpha_{i,j}$: Attention weights, e.g. $\alpha_{i,j} = \dfrac{\exp(\vec{a}^T[W\vec{\mathrm{x}}_i, W\vec{\mathrm{x}}_{\mathrm{j}}])}{\sum_{\mathrm{k} \in \mathrm{N(i)}} \exp(\vec{a}^T[W\vec{\mathrm{x}}_i, W\vec{\mathrm{x}}_{\mathrm{k}}])}$

Spectral Methods: Spectral « translation » seems to be a translation

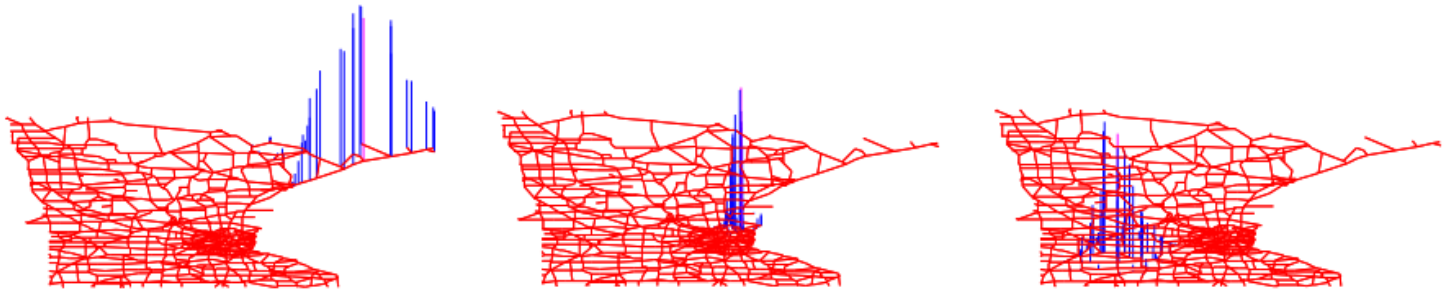
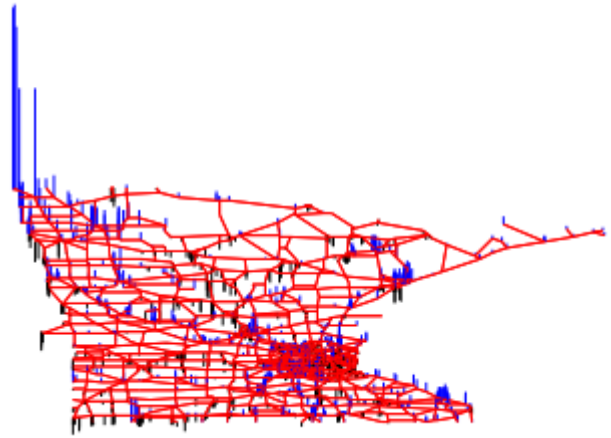


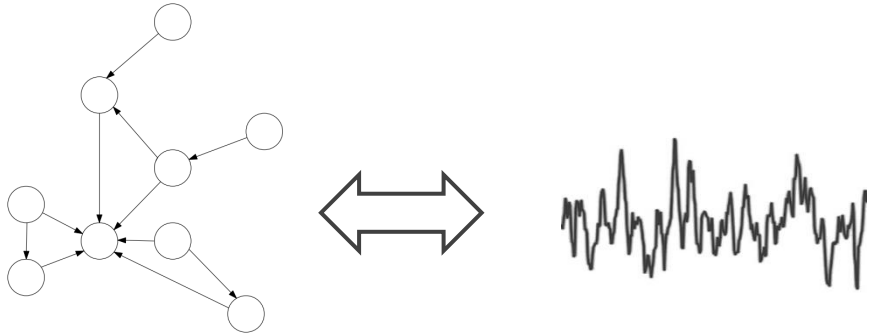
Fig. 2. The translated signals (a) $T_{200}f$, (b) $T_{1000}f$, and (c) $T_{2000}f$, where f, the signal shown in Figure 1(c), is a normalized heat kernel satisfying $\hat{f}(\ell) = Ce^{-5\lambda_\ell}$.



From Shuman, Ricaud, Vandergheynst (2012):
A Windowed Graph Fourier Transform

4. Spectral Graph Convolutional Networks

Localizing spectral graph convolution:



ency domain

- 'Frequency domain': Change of basis, $\hat{x} = U^T \vec{x}$; $\vec{x} = U U^T \vec{x}$

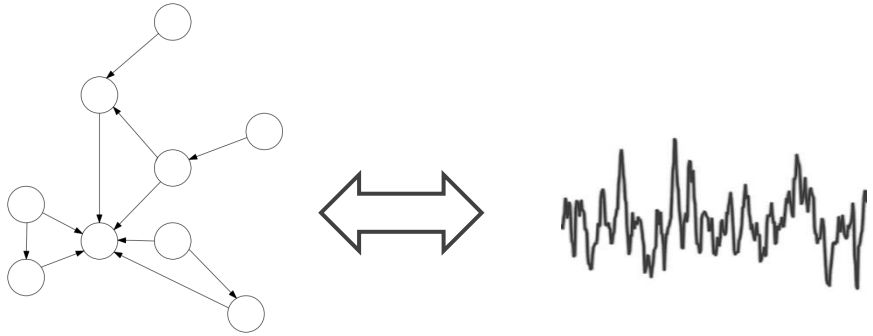
→ « Convolution » on a graph:

$$\vec{u}_i \rightarrow \theta_i \vec{u}_i$$

$$\vec{x}_{l+1} = U \, diag(\vec{\theta}) \, U^T \vec{x}_l$$

4. Spectral Graph Convolutional Networks

Localizing spectral graph convolution:



ency domain

- 'Frequency domain': Change of basis, $\hat{x} = U^T \vec{x}$; $\vec{x} = UU^T \vec{x}$

→ « Convolution » on a graph:

$$\vec{u}_i \rightarrow \theta_i \vec{u}_i$$

$$\vec{x}_{l+1} = \boldsymbol{L^k}\ \vec{x}_l$$ (simplified)

$\boldsymbol{L}$: Graph Laplacian,
combines 1-hop neighbours

CoNLL-2009: Syntactic and semantic trees

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/35497.pdf

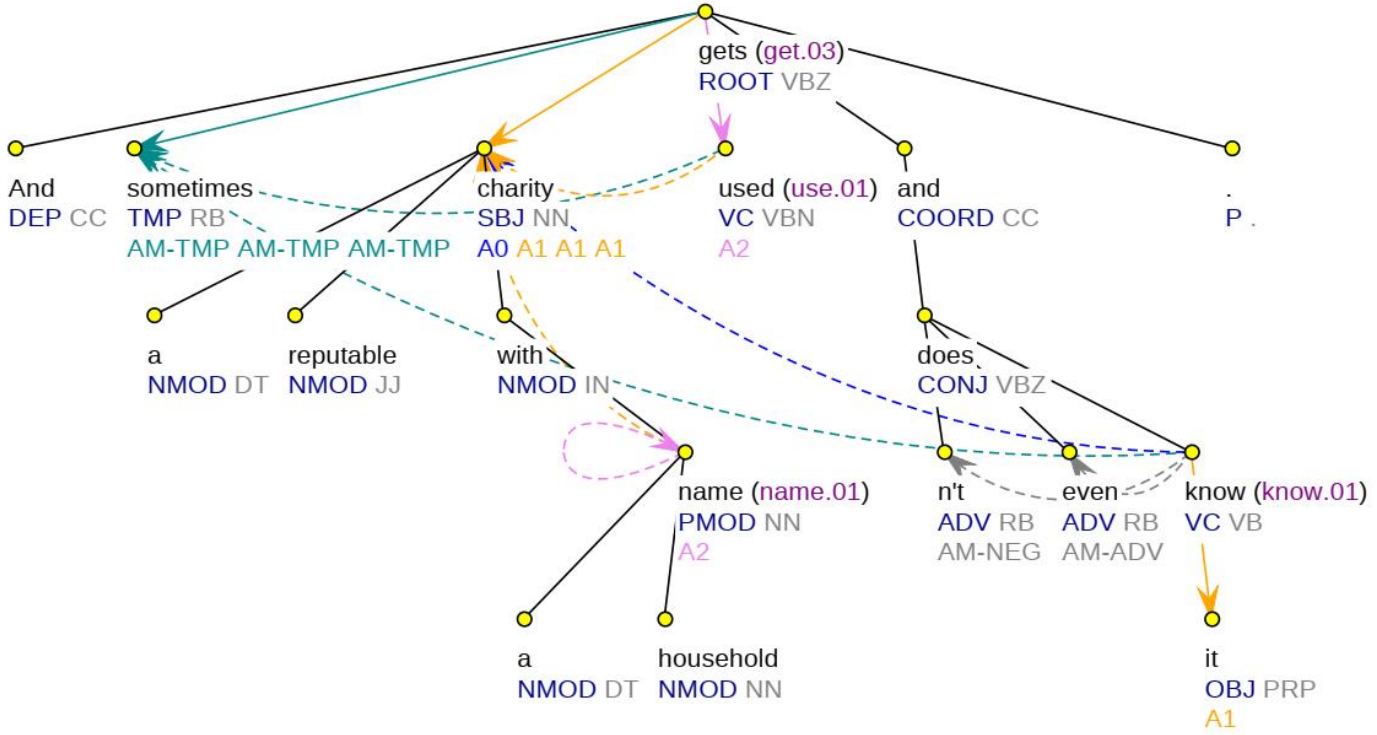


Figure 1: Visualisation of the English sentence *"And sometimes a reputable charity with a houshold name gets used and doesn't even know it."* (Penn Treebank, wsj_0559) showing jointly the labeled syntactic and semantic dependencies. The basic tree shape comes from the syntactic dependencies; syntactic labels and POS tags are on the 2^{nd} line at each node. Semantic dependencies which do not follow the syntactic ones use dotted lines. Predicate senses in parentheses (`use:01`, ...) follow the word label. SRLs (`A0`, `AM-TMP`, ...) are on the last line. Please note that multiple semantic dependencies (e.g., there are four for *charity*: A0 ← *know*, A1 ← *gets*, A1 ← *used*, A1 ← *name*) and self-dependencies (*name*) appear in this sentence.

4. Spectral Graph Convolutional Networks

What basis U ?

Graph Laplacian: $\boldsymbol{L = D - A}$

Or: W, W_{ij}: edge weights

A: Adjacency matrix

D: diag(D_{ii}), $D_{ii} = \sum_j A_{ij}$, node degree

$L = U \Lambda U^T$, U: Eigenvectors of L

For graph representation learning:

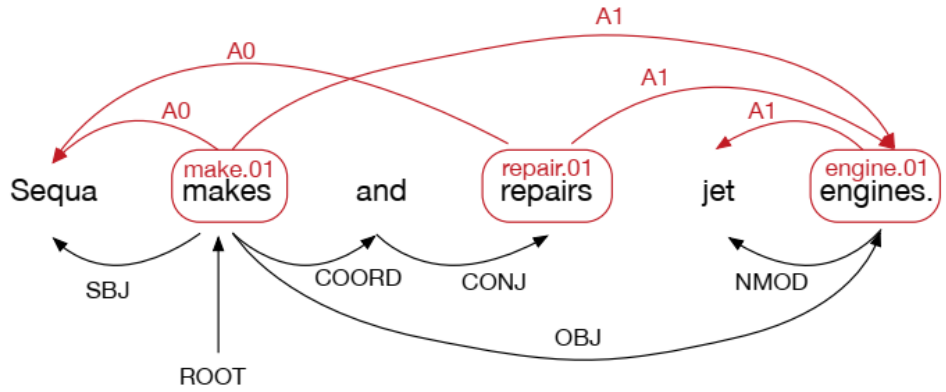
2015: TransR: https://ojs.aaai.org/index.php/AAAI/article/view/9491/9350

2018: https://arxiv.org/pdf/1706.02216.pdf
Inductive Representation Learning on Large Graphs

Example: Semantic role labelling

- Given: a sentence and its syntax tree (in black)
- Determine “who did what to whom” (in red)

[9]: BiLSTM
+ gated syntax-GCN



https://arxiv.org/pdf/1703.04826.pdf