UNIVERSITÄT BIELEFELD

Faculty of Technology

CORE Lab

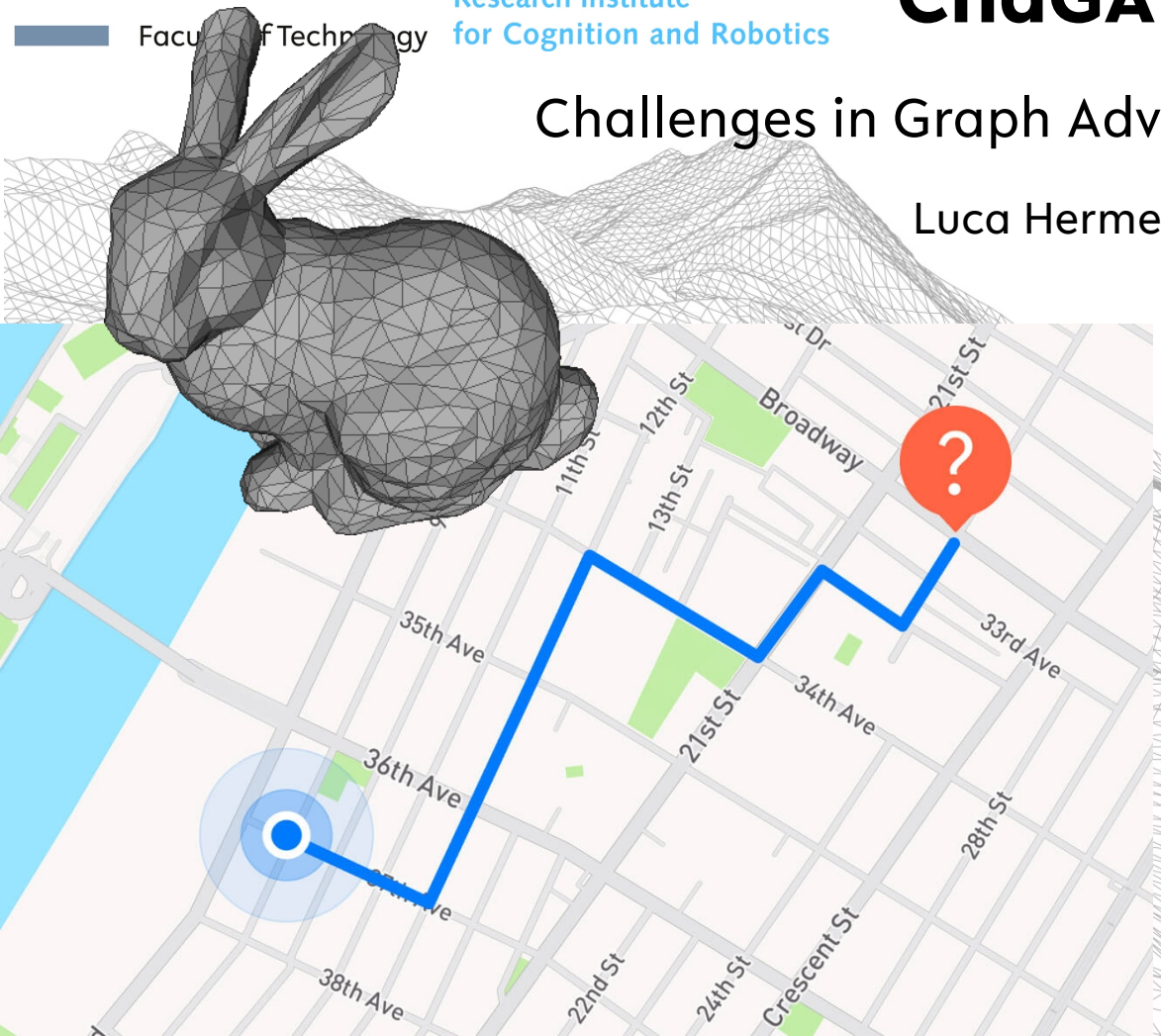
Research Institute for Cognition and Robotics

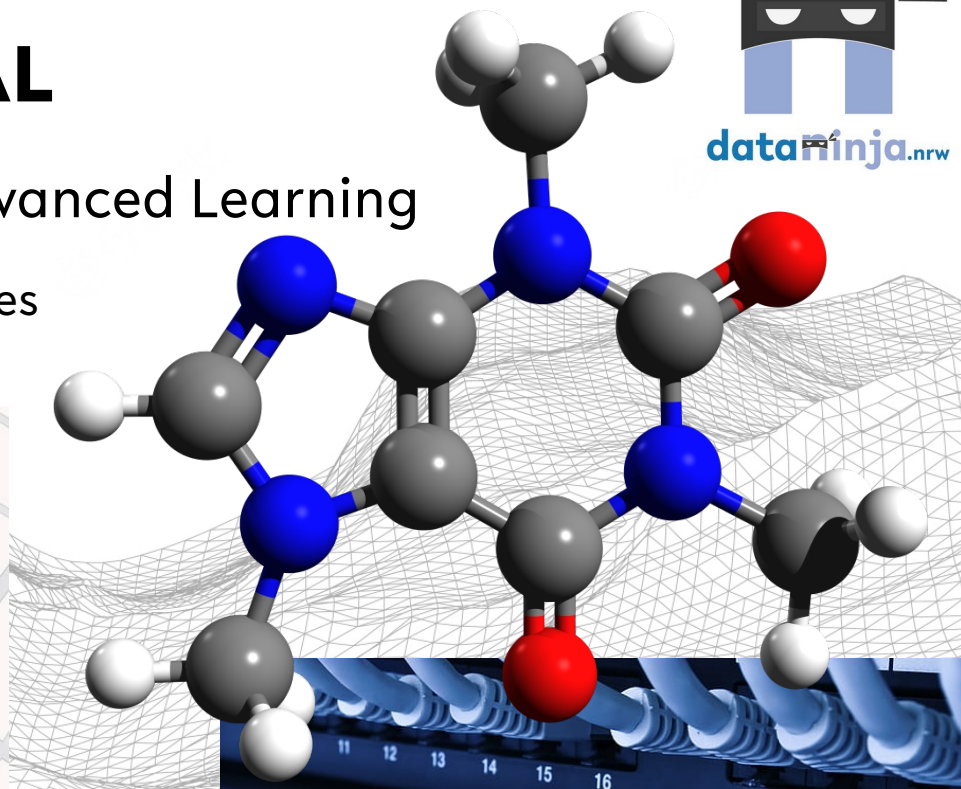
ChaGAL

Challenges in Graph Advanced Learning

Luca Hermes

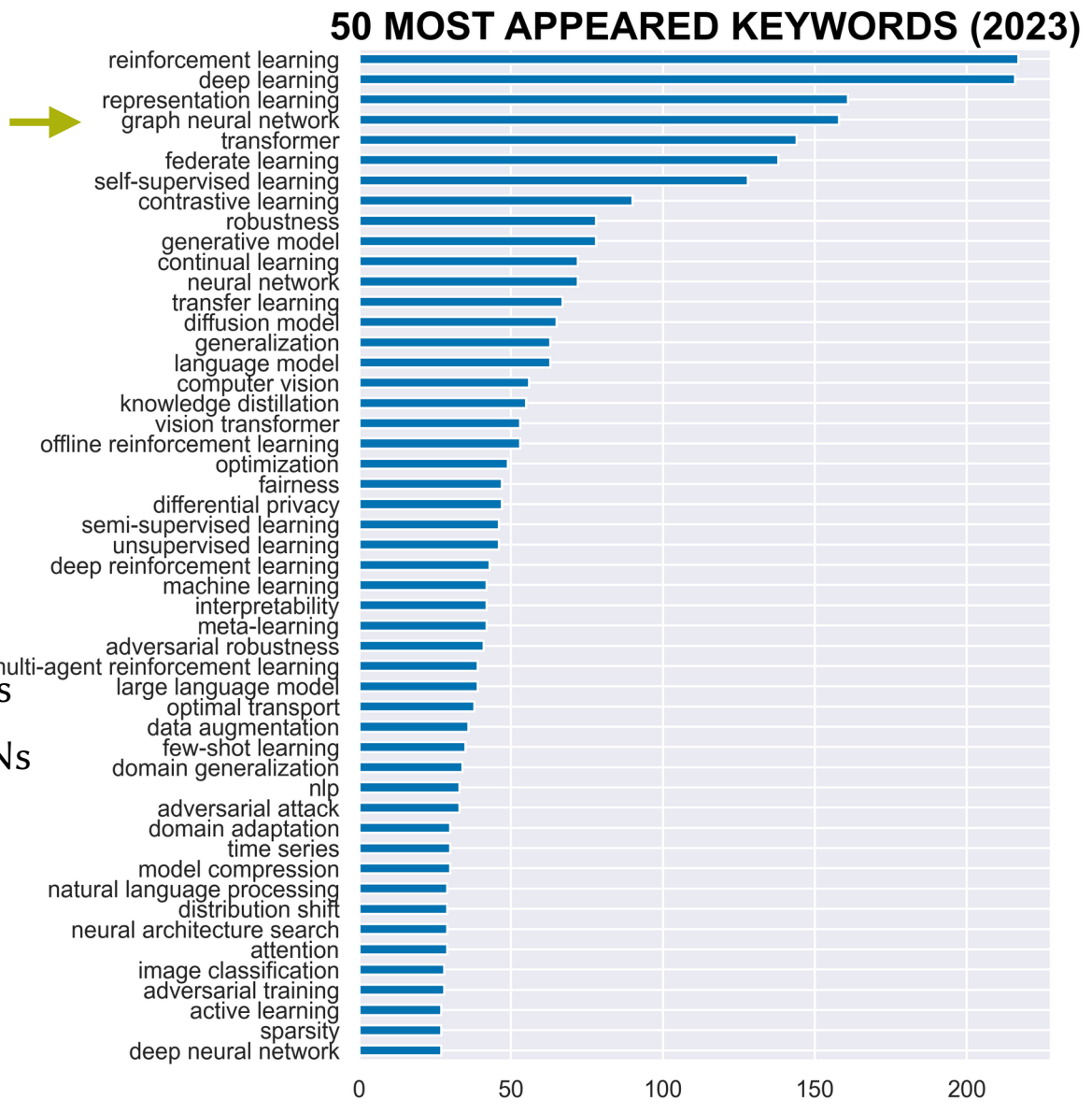
dataNinja.nrw





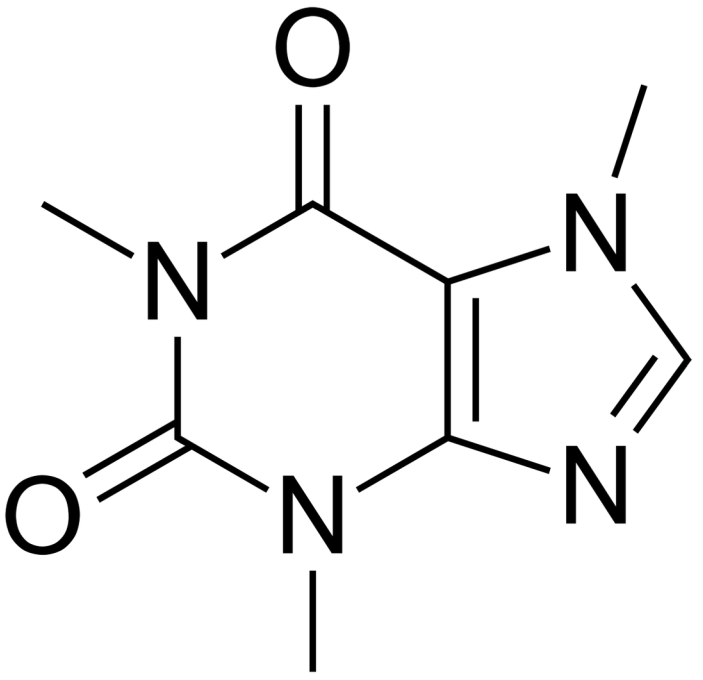
Outline

- Graphs as a Data Structure
- Introduction to Graph Neural Networks
  - How to represent structure
  - GNNs are local operations
- ConvNets are specialized grid GNNs
- Aggregation can lead to Oversmoothing
- Case Study: Water Distribution Networks
- My current project: Sampling-based GNNs
- Benchmarking GNNs
  A topic often overlooked



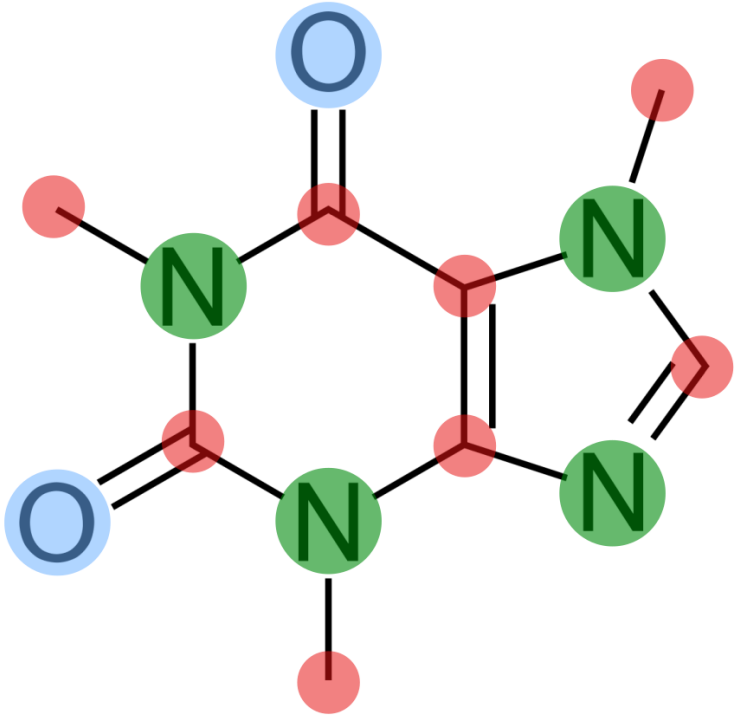
What are Graphs?

- A Graph G consists of a set of nodes V and edges E
  G = (V, E)



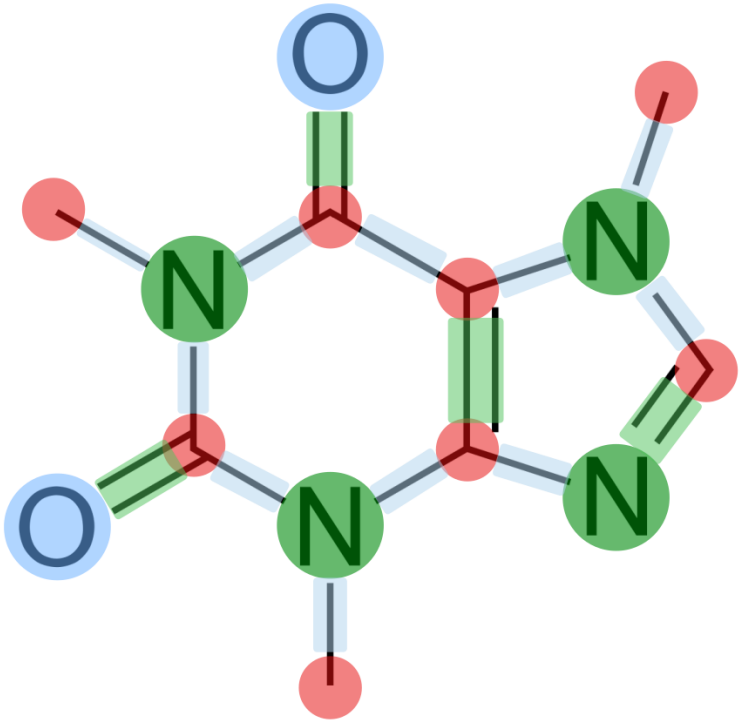
What are Graphs?

- A Graph G consists of a set of nodes V and edges E
  G = (V, E)
- Nodes V are specified by feature vectors $\mathbf{v} \in \mathbb{R}^{d_v}$
- An edge connects, or *relates*, two nodes



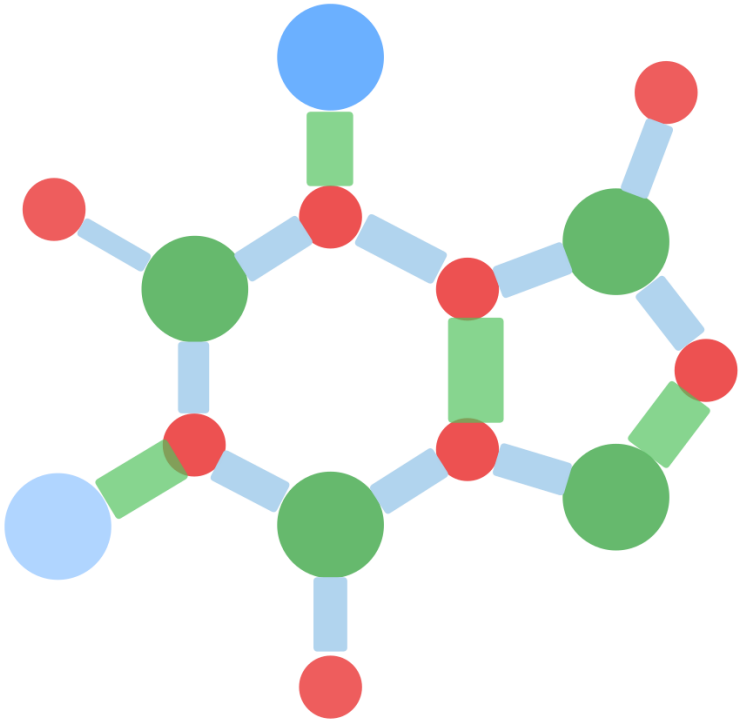
What are Graphs?

- A Graph G consists of a set of nodes V and edges E
  G = (V, E)
- Nodes V are specified by feature vectors $\mathbf{v} \in \mathbb{R}^{d_v}$
- An edge connects, or *relates*, two nodes
- Edges E may be specified by feature vectors $\mathbf{e} \in \mathbb{R}^{d_e}$

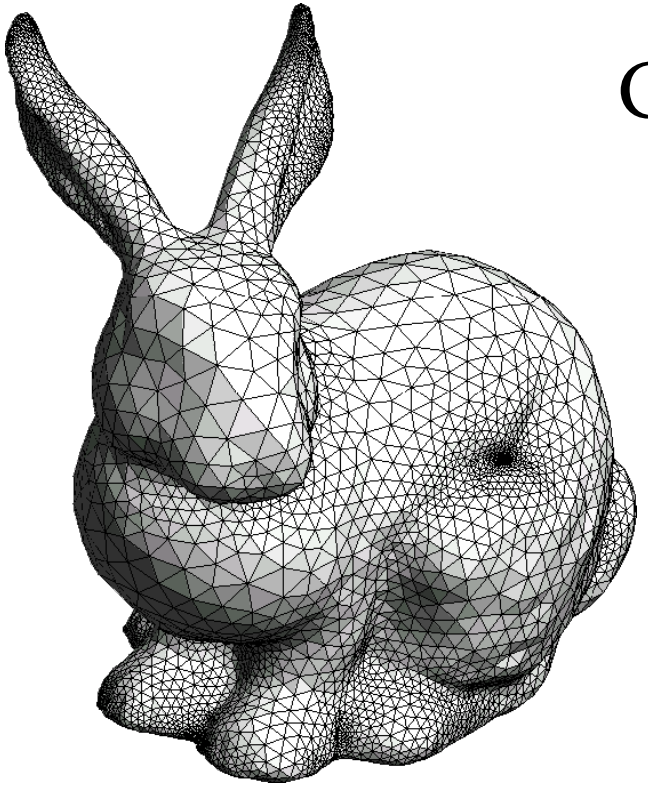


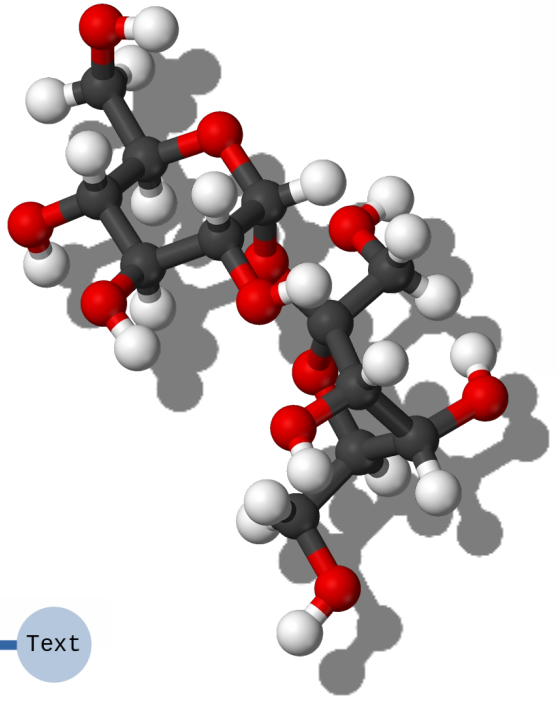
What are Graphs?

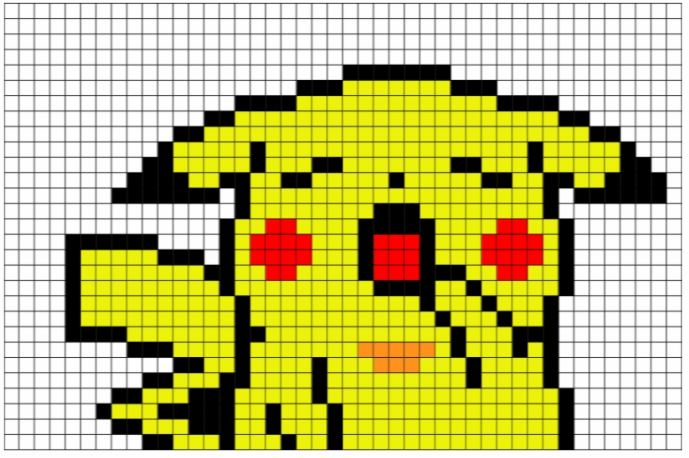
- A Graph G consists of a set of nodes V and edges E
  G = (V, E)
- Nodes V are specified by feature vectors $\mathbf{v} \in \mathbb{R}^{d_v}$
- An edge connects, or *relates*, two nodes
- Edges E may be specified by feature vectors $\mathbf{e} \in \mathbb{R}^{d_e}$



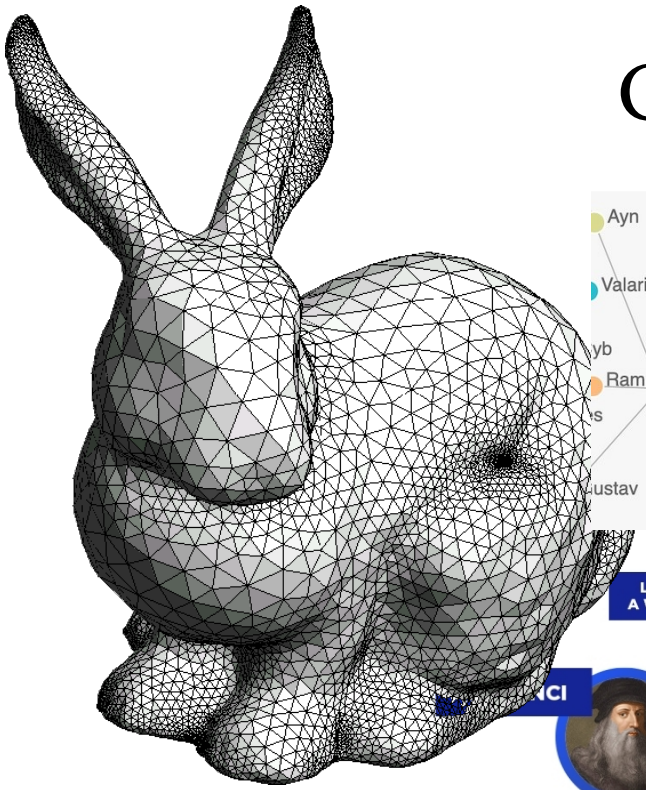
Graphs are Everywhere

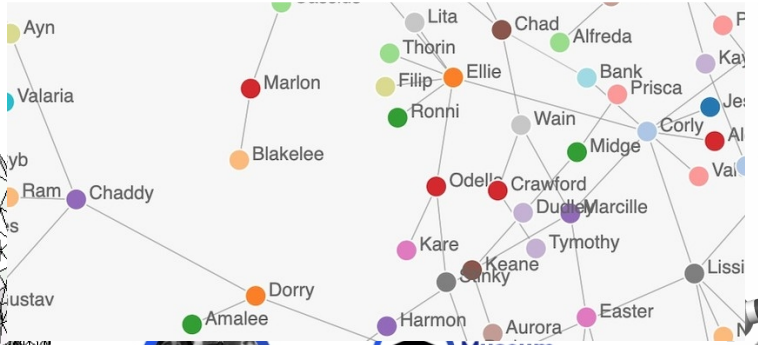


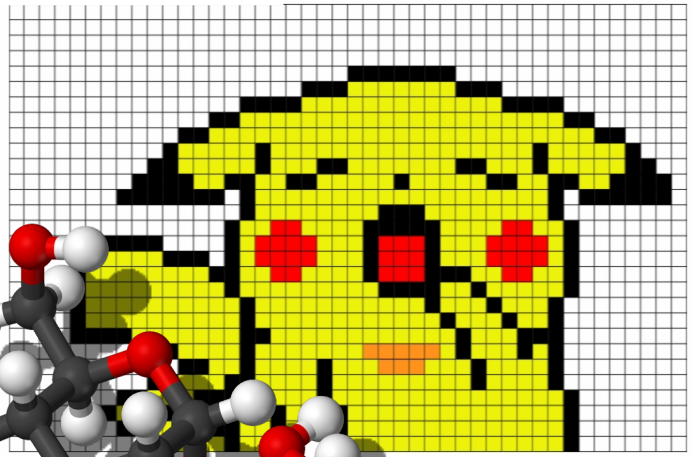


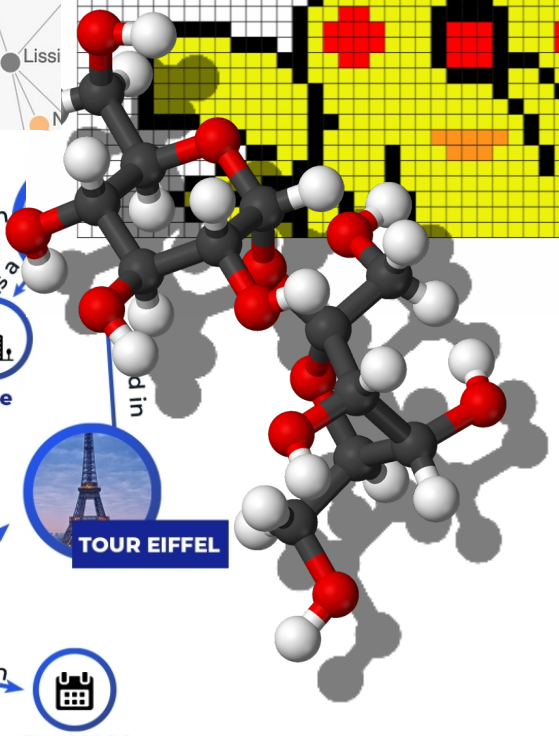


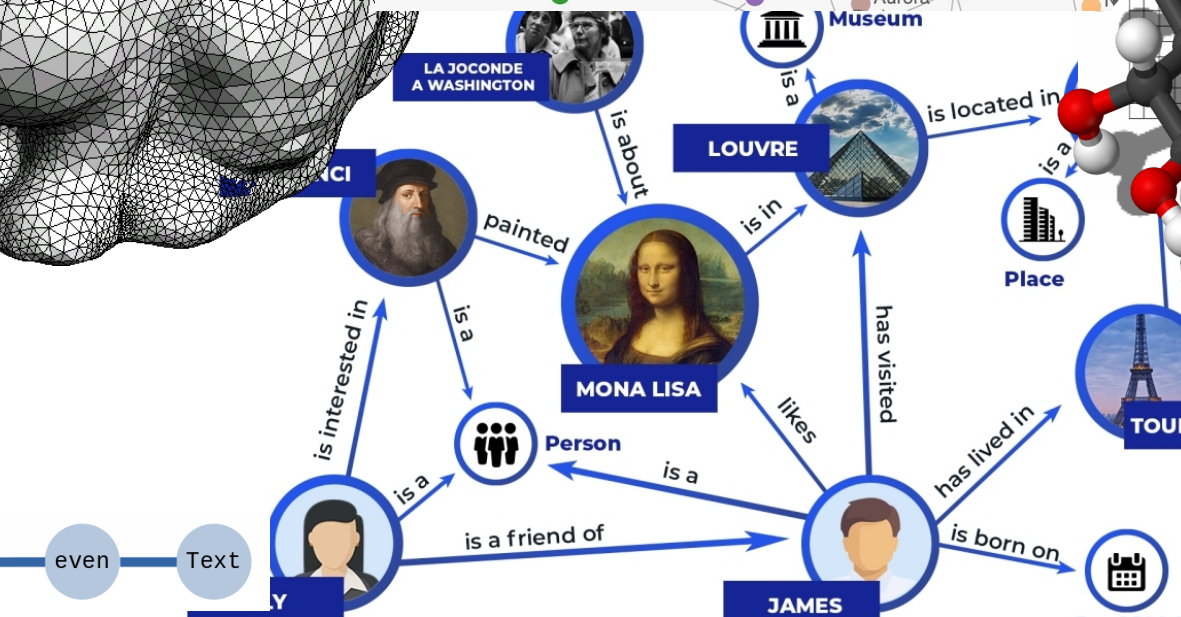
Graphs are Everywhere

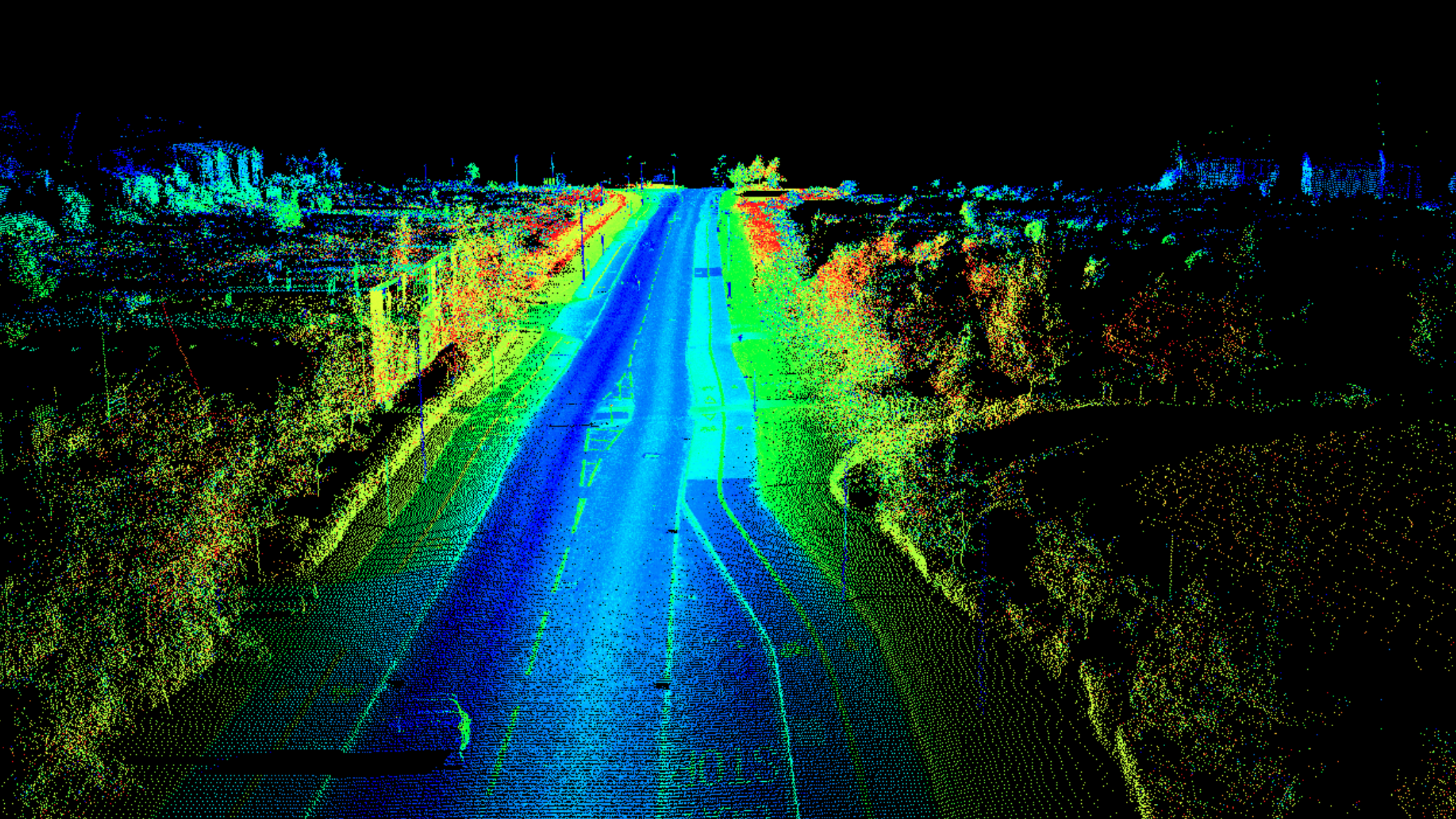










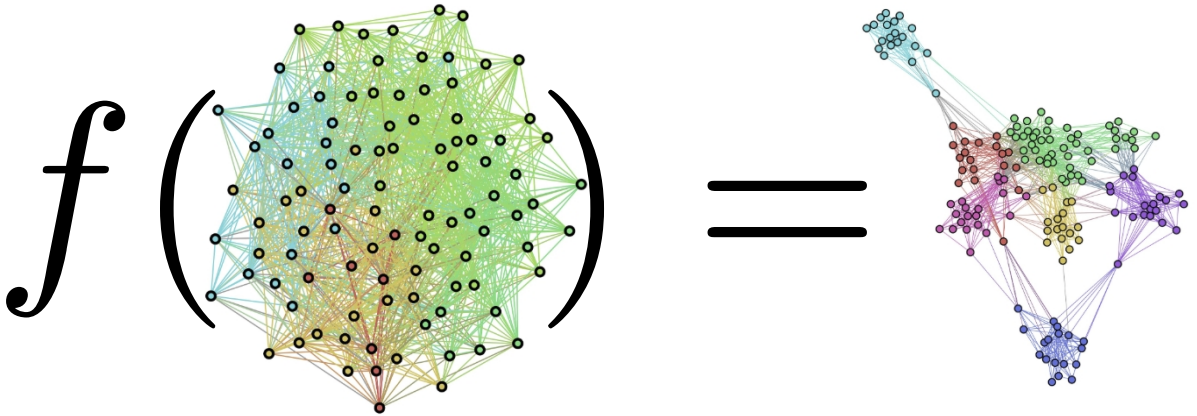


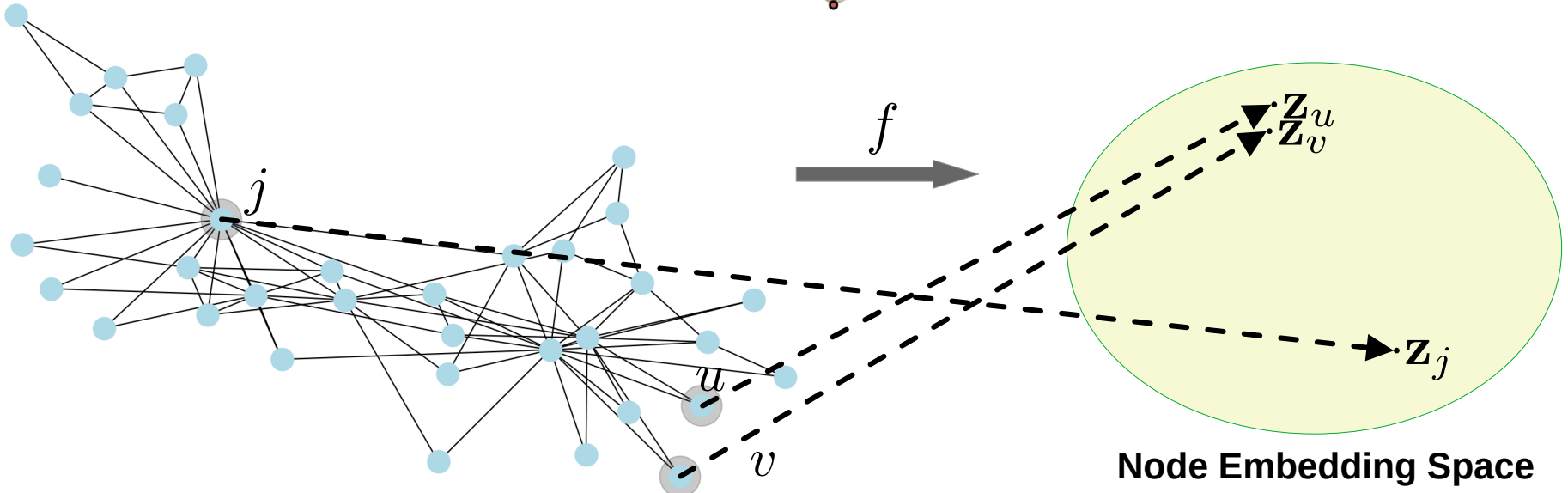
When the data contains multiple manifestations of similar things and these things are relatable, then it is a graph:

	Thing		Relation
Images	Pixels	–	Proximity
Text	Words	–	Context/Proximity
Molecules	Atoms	–	Bonds
Ontologies	Subject/Object	–	Predicate
Social	User	–	Relationship
Point Cloud	3D-Point	–	Proximity
Research	Paper	–	Citation

What are Graph Neural Networks?

- Functions that embeds nodes based on structure and node features
- **Two nodes in a similar structural context should be mapped to similar locations**



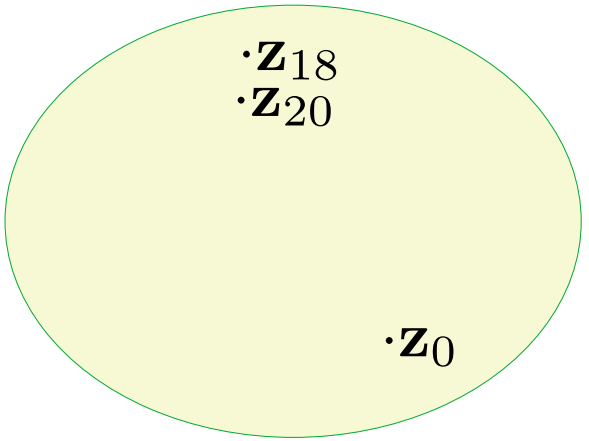


- **Two nodes in a similar structural context should be mapped to similar locations**

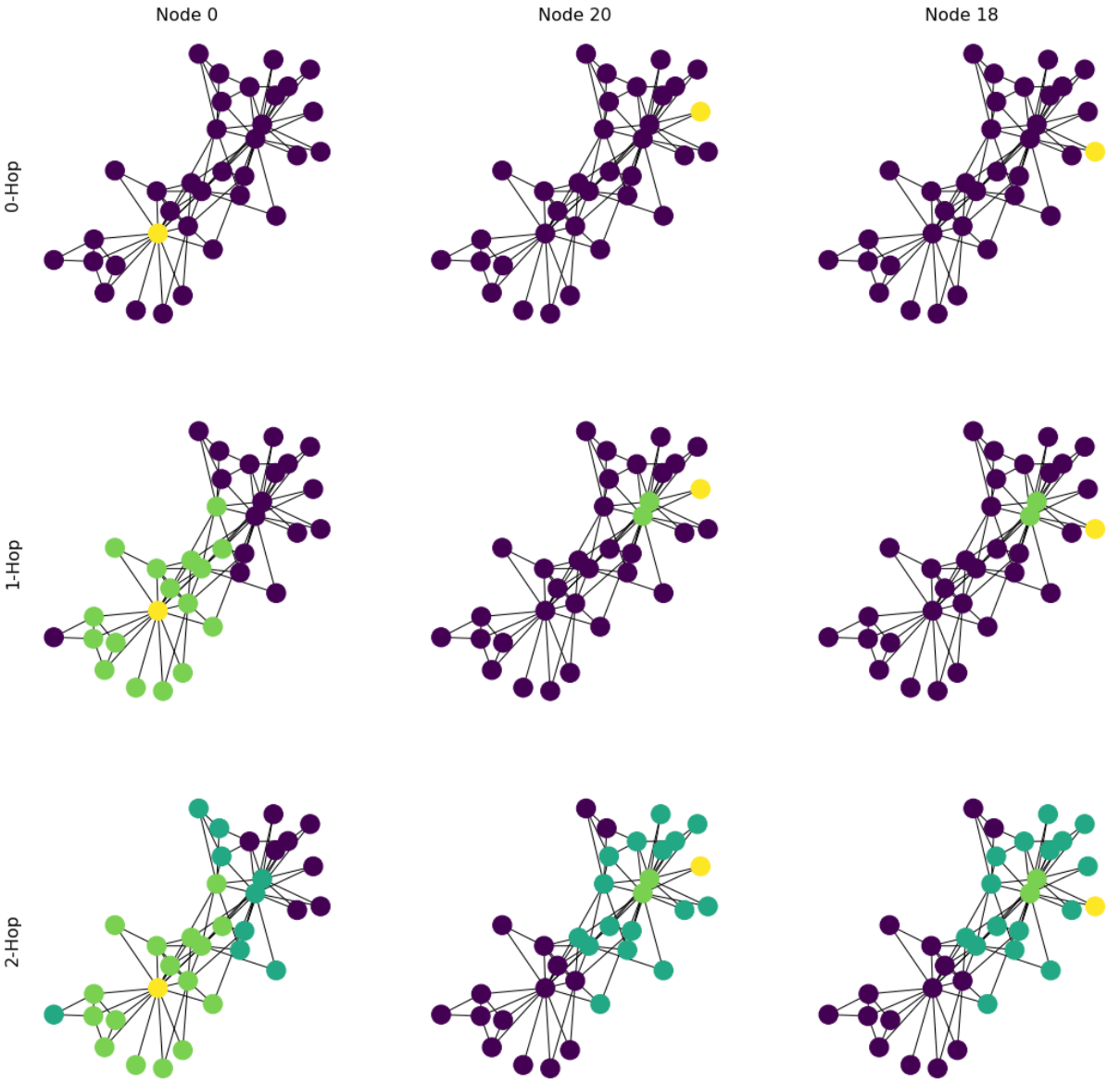
  → This is equivalent to discriminating subgraphs

- Node 20 and 18 have a similar subgraph, they should be close in the latent space

  → **How to compare these subgraphs?**



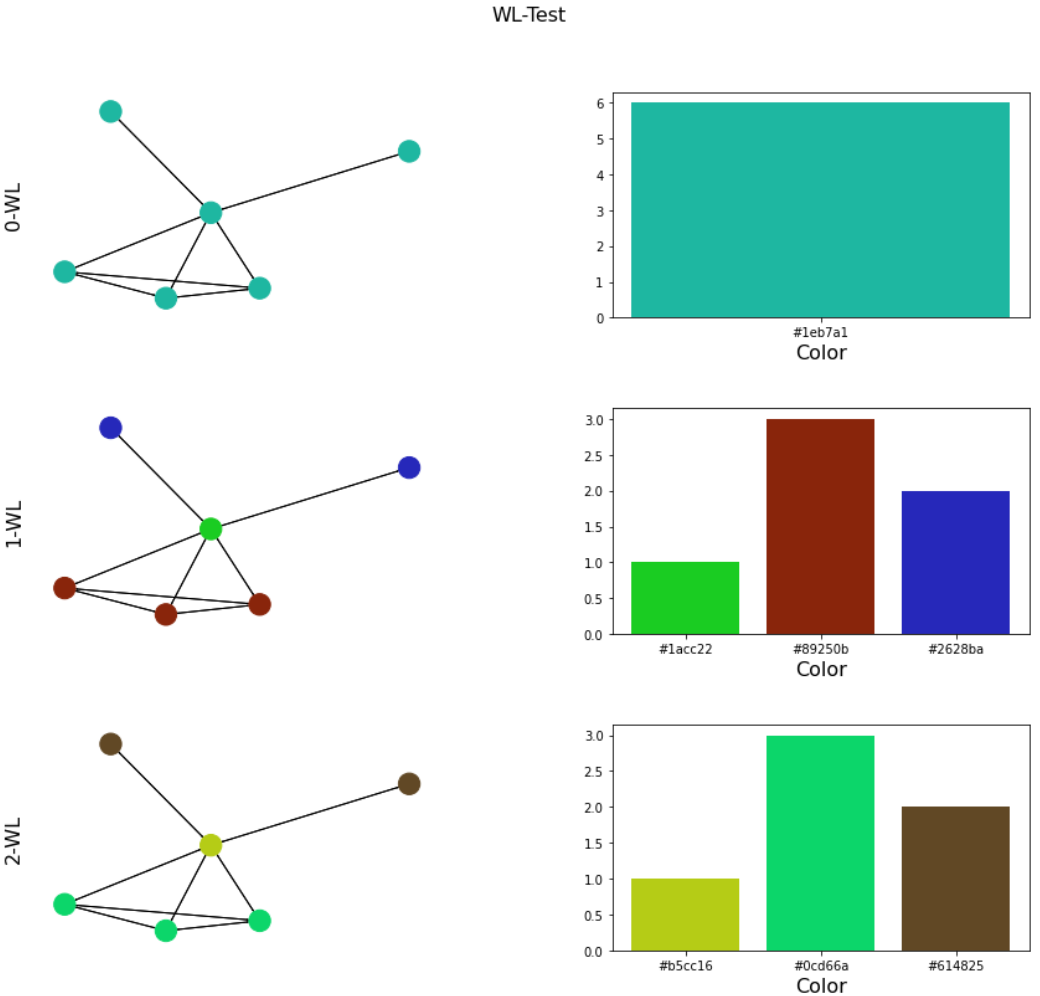
d-dimensional Latent space



The Weisfeiler-Lehman Test

- Canonically represents structure
- Algorithm:

Initialization: Color every node similarly
$\mathbf{c} \leftarrow \mathbf{1}$
for i iterations :
  for each node $n \in N$:
    Hash the multiset $u \in \mathcal{N}_n$
    of neighboring nodes
    $c_n \leftarrow \texttt{HASH}\,(u \in \mathcal{N}_n)$



B. Weisfeiler, A. Lehman: The reduction of a graph to canonical form and the algebra which appears therein (1968)

The Weisfeiler-Lehman Test

- Canonically represents structure
- Algorithm:

Initialization: Color every node similarly

$\mathbf{c} \leftarrow \mathbf{1}$

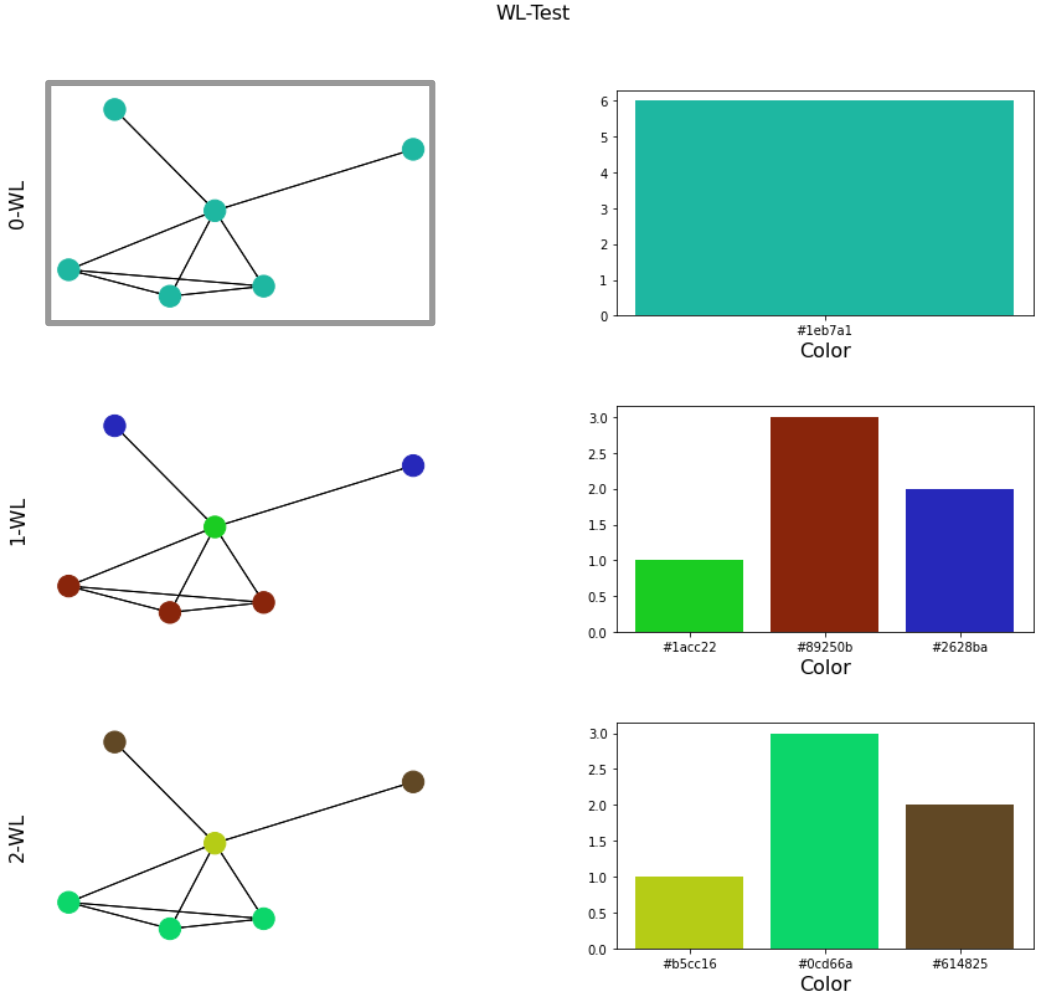
for i iterations :

  for each node $n \in N$:

    Hash the multiset $u \in \mathcal{N}_n$

    of neighboring nodes

    $c_n \leftarrow \texttt{HASH}\,(u \in \mathcal{N}_n)$



B. Weisfeiler, A. Lehman: The reduction of a graph to canonical form and the algebra which appears therein (1968)

The Weisfeiler-Lehman Test

- Canonically represents structure
- Algorithm:

Initialization: Color every node similarly

$\mathbf{c} \leftarrow \mathbf{1}$

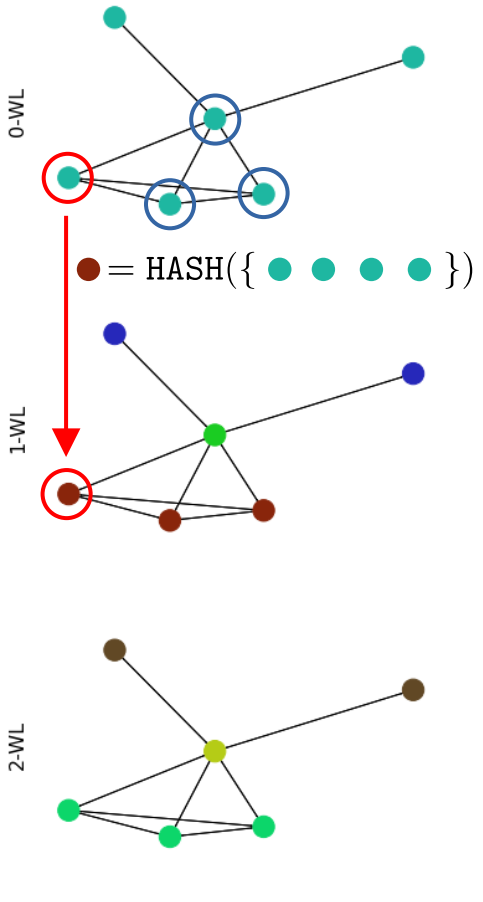
for i iterations :

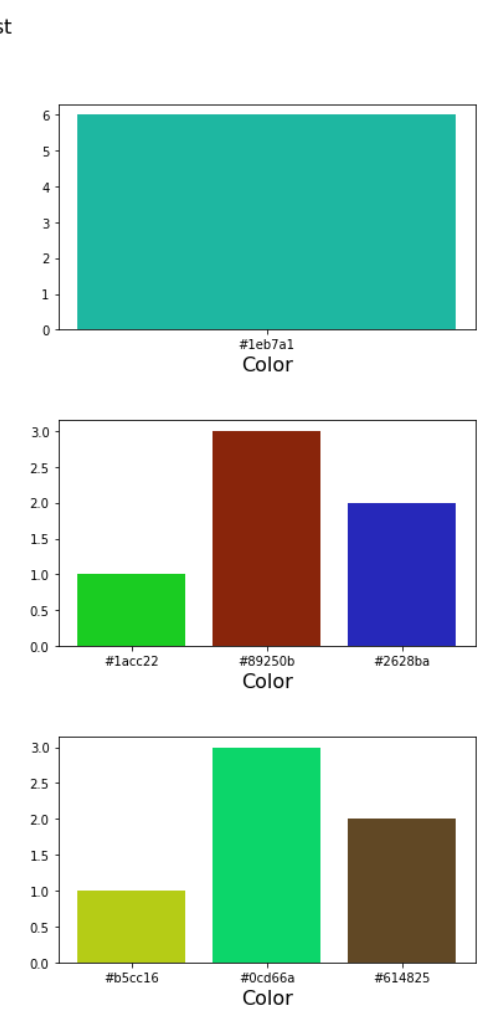
  for each node $n \in N$:

    Hash the multiset $u \in \mathcal{N}_n$

    of neighboring nodes

$$c_n \leftarrow \texttt{HASH}\,(u \in \mathcal{N}_n)$$





B. Weisfeiler, A. Lehman: The reduction of a graph to canonical form and the algebra which appears therein (1968)

The Weisfeiler-Lehman Test

- Canonically represents structure
- Algorithm:

Initialization: Color every node similarly

$\mathbf{c} \leftarrow \mathbf{1}$

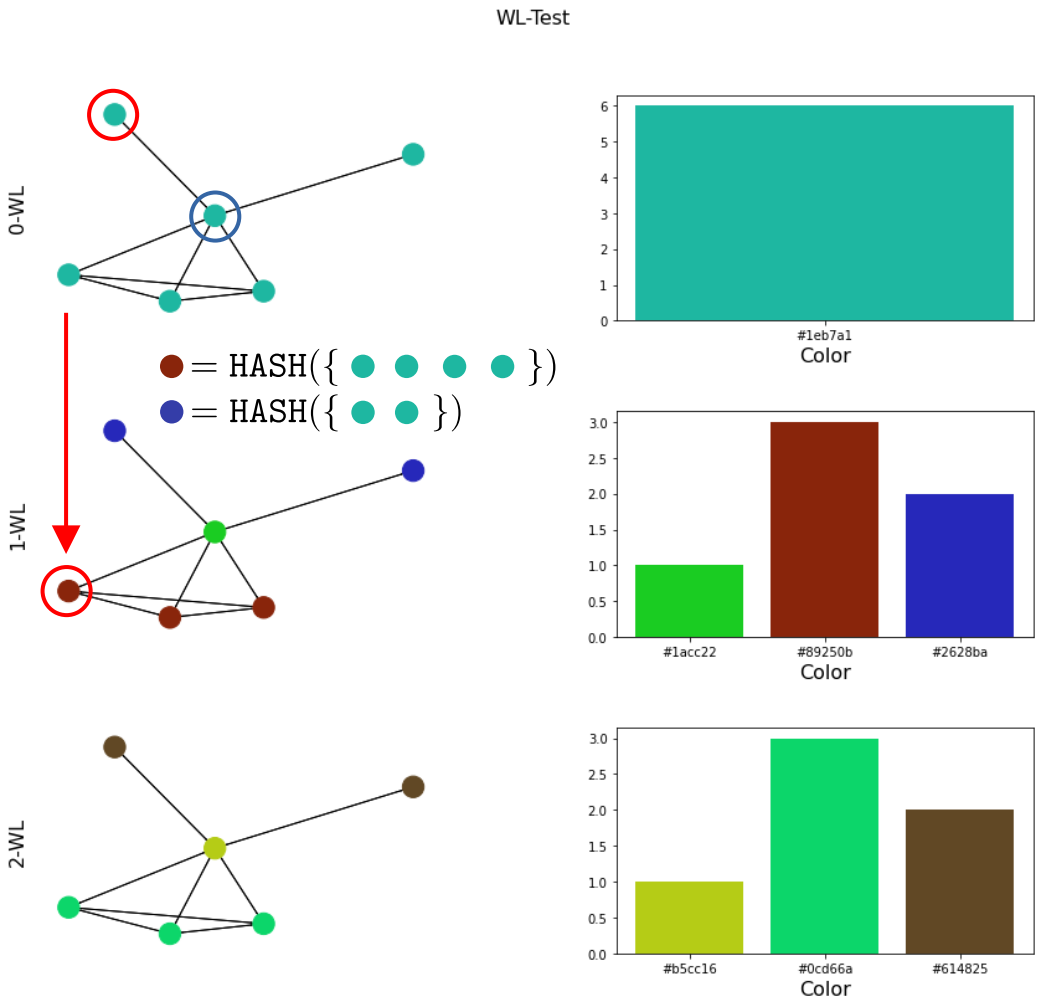
for i iterations :

  for each node $n \in N$:

    Hash the multiset $u \in \mathcal{N}_n$

    of neighboring nodes

$$c_n \leftarrow \texttt{HASH}\,(u \in \mathcal{N}_n)$$



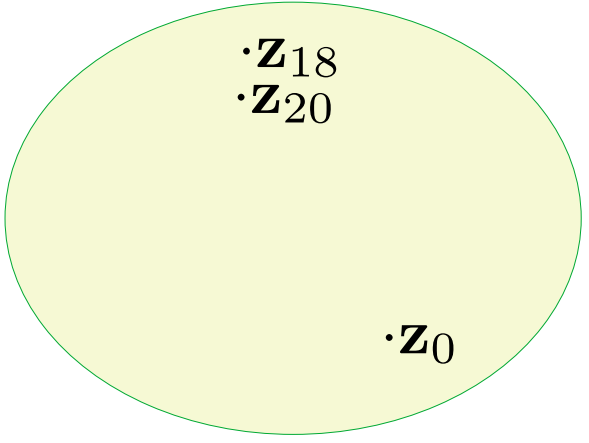
B. Weisfeiler, A. Lehman: The reduction of a graph to canonical form and the algebra which appears therein (1968)

- **Two nodes in a similar structural context should be mapped to similar locations**

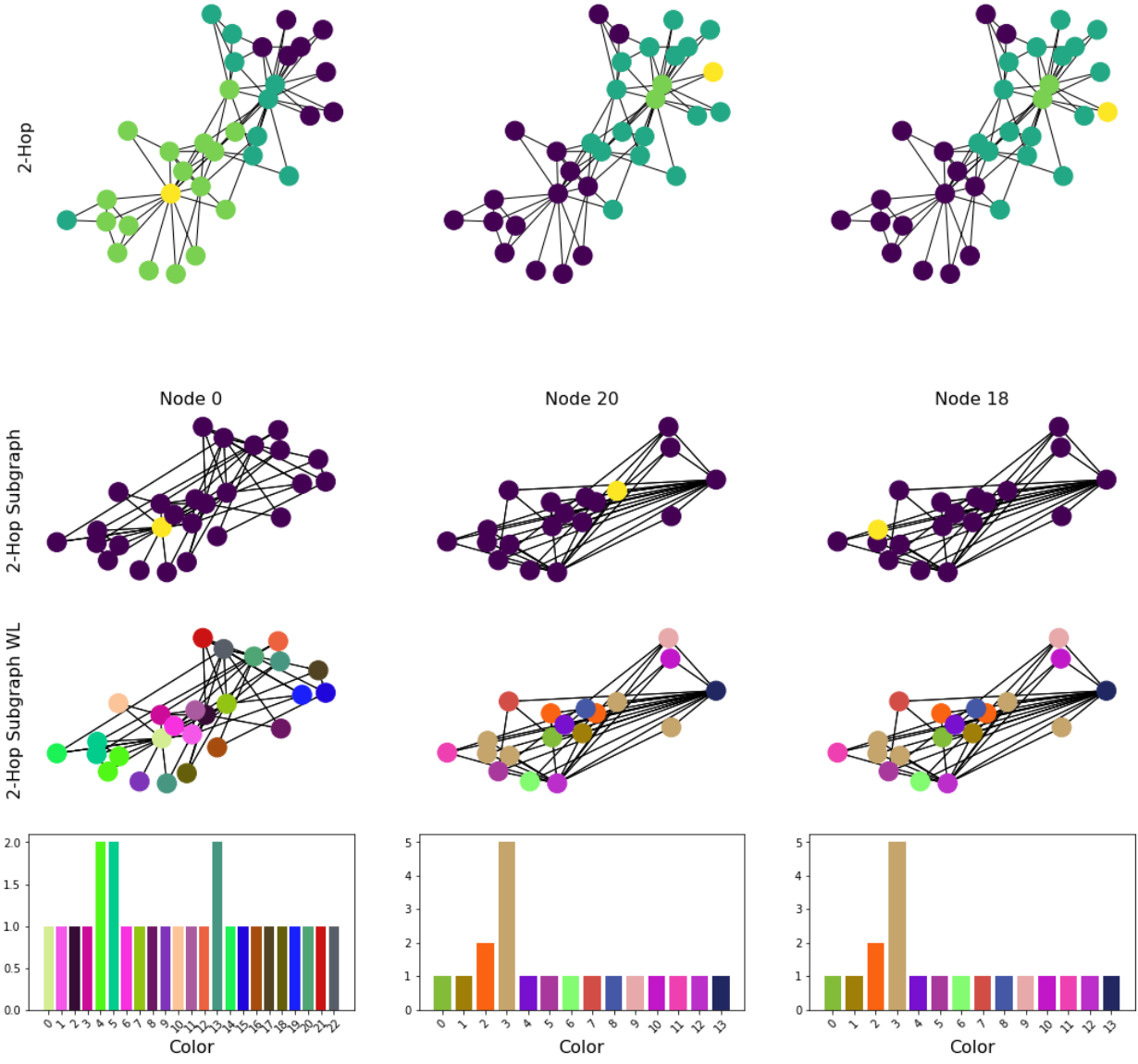
  → This is equivalent to discriminating subgraphs

- Node 20 and 18 have a similar subgraph, they should be close in the latent space

  → **How to compare these subgraphs?**



d-dimensional Latent space

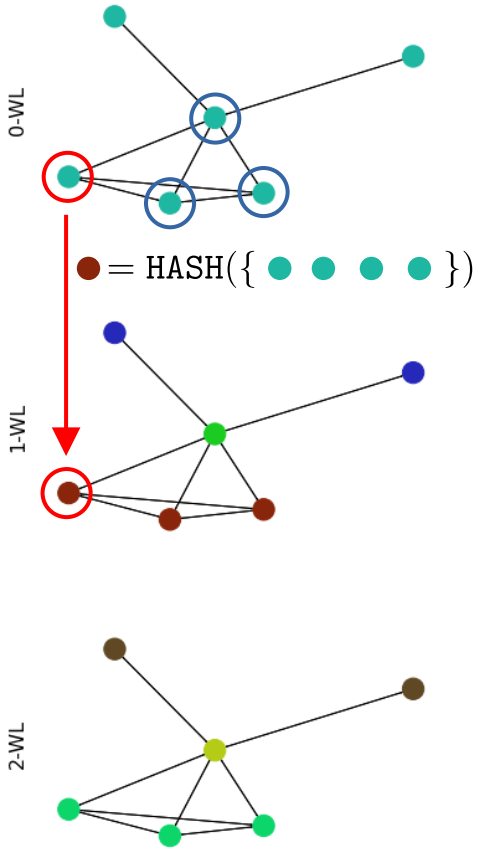


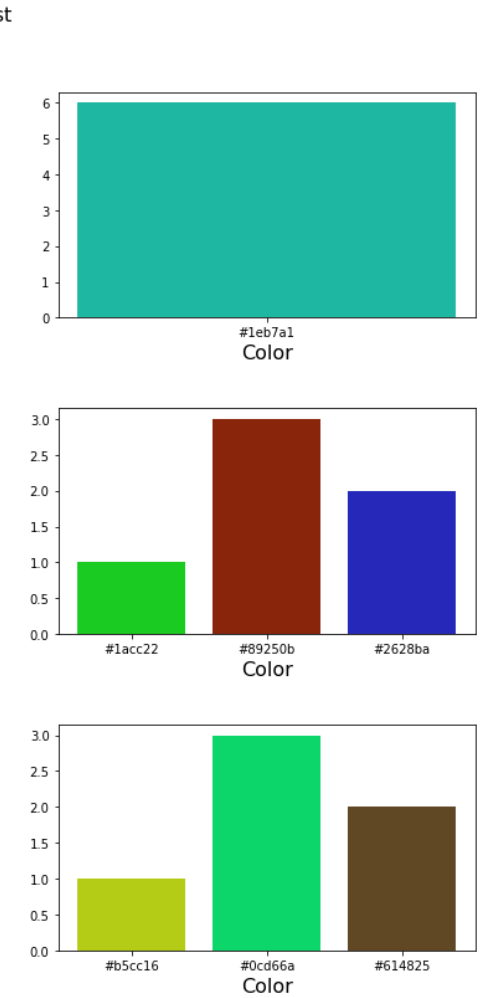
WL-Test - Graph Convolution

- A Graph Convolution (GCN) is a differentiable version of the WL-Algorithm
- Instead of a hash function, a GCN applies set aggregation and a consecutive MLP

WL node update: $\bullet = \texttt{HASH}(\{ \bullet \bullet \bullet \bullet \})$

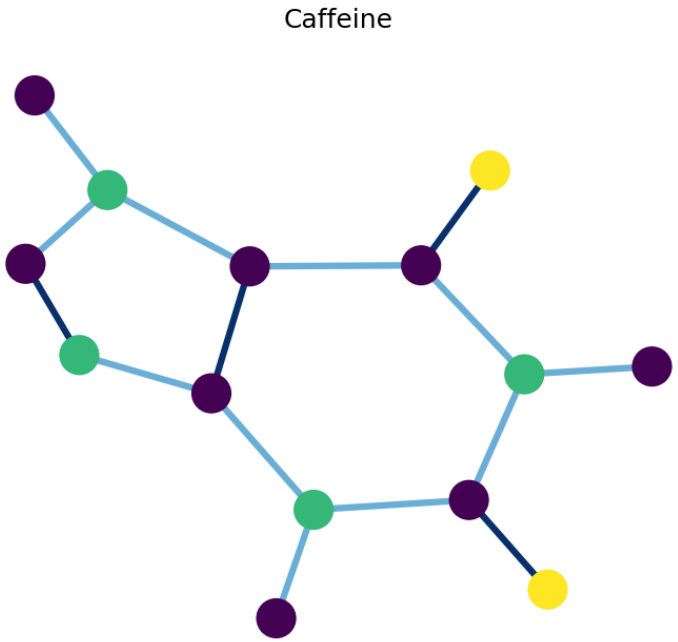
GCN node update: $n' = \texttt{MLP}_\Phi(\sum_{u \in \mathcal{N}_n} u)$



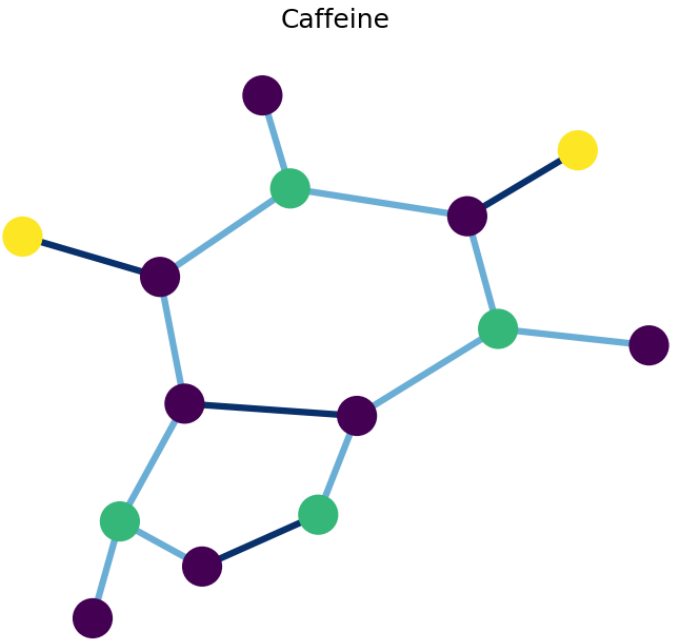


Kipf et al.: Semi-Supervised Classification with Graph Convolutional Networks

From Set-Hash to Set-MLP



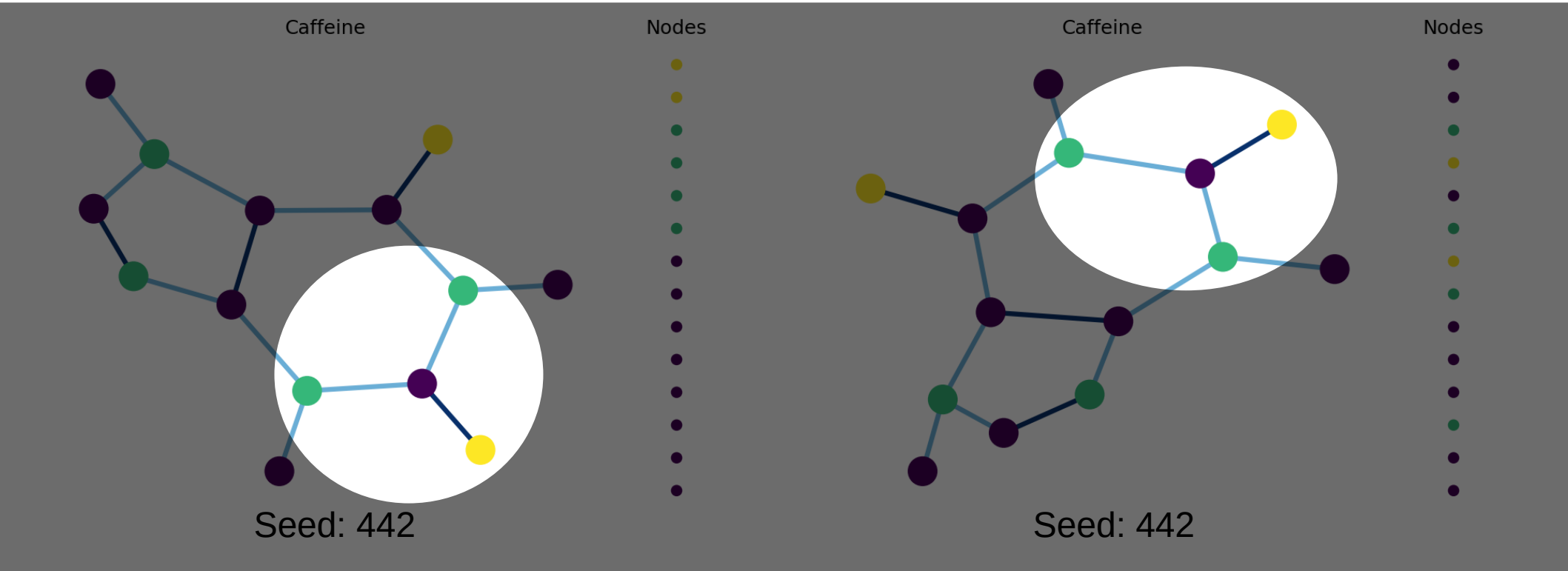
Seed: 442



Seed: 442

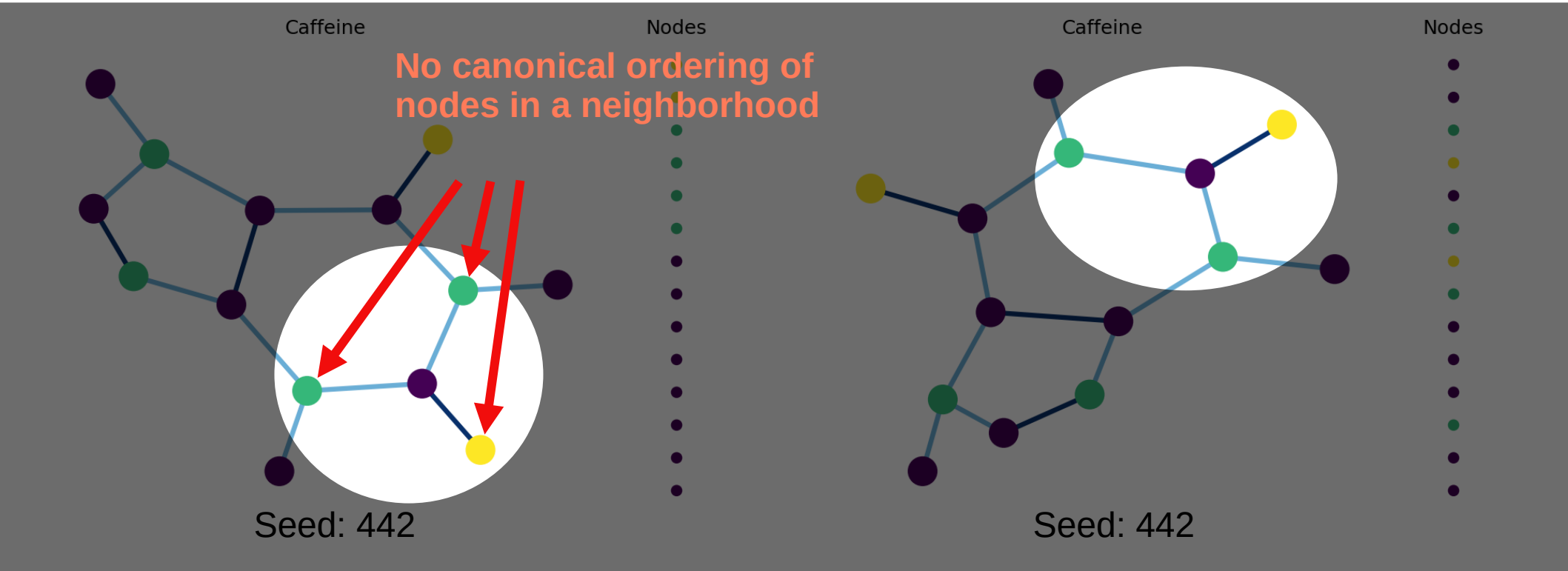
What happens locally in a GNN?

- How to replace the Hash function with an MLP



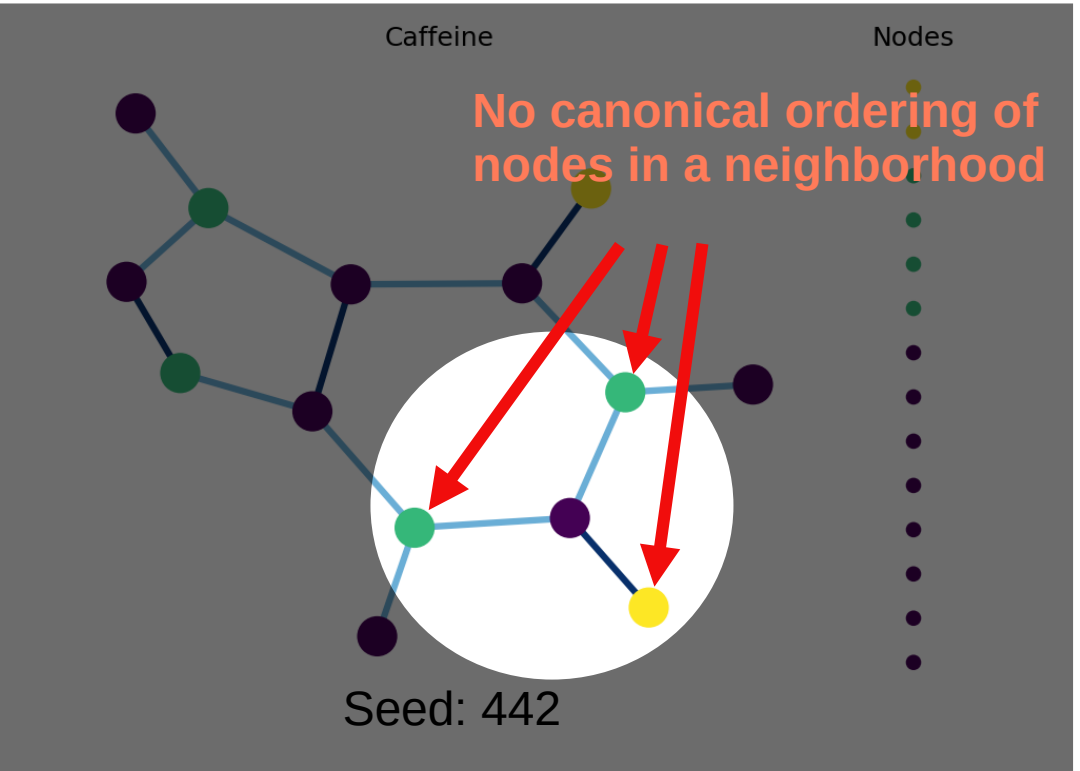
What happens locally in a GNN?

- Node neighborhoods remain the same under permutation



What happens locally in a GNN?

- Node neighborhoods remain the same under permutation



How are we encoding these neighborhoods?

With a permutation invariant function, i.e. a function that satisfies:

$$f : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{d} \qquad f(X) = f(PX)$$

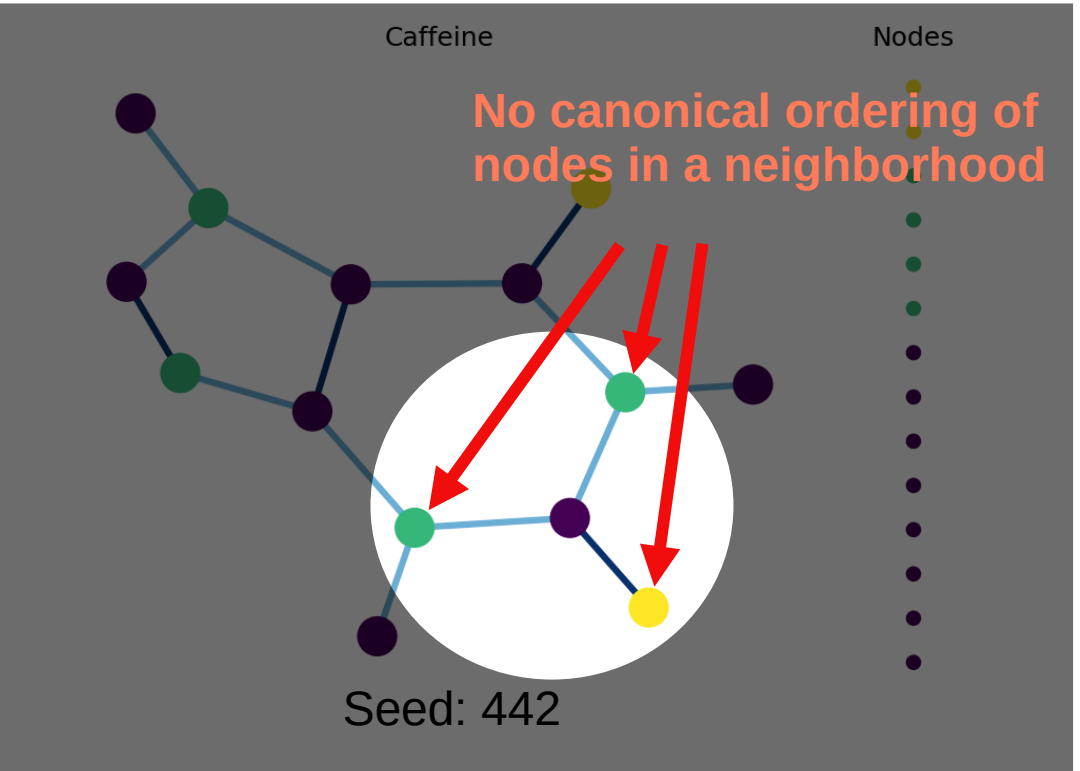
$$f\left(\begin{array}{c}\bullet\\\bullet\\\bullet\\\bullet\end{array}\right) = f\left(\begin{array}{c}\bullet\\\bullet\\\bullet\\\bullet\end{array}\right)$$

Permutation Matrix

Neighborhood Node Set

What happens locally in a GNN?

- Node neighborhoods remain the same under permutation



Possible Perm. Invariant Functions that map a Set onto a vector:

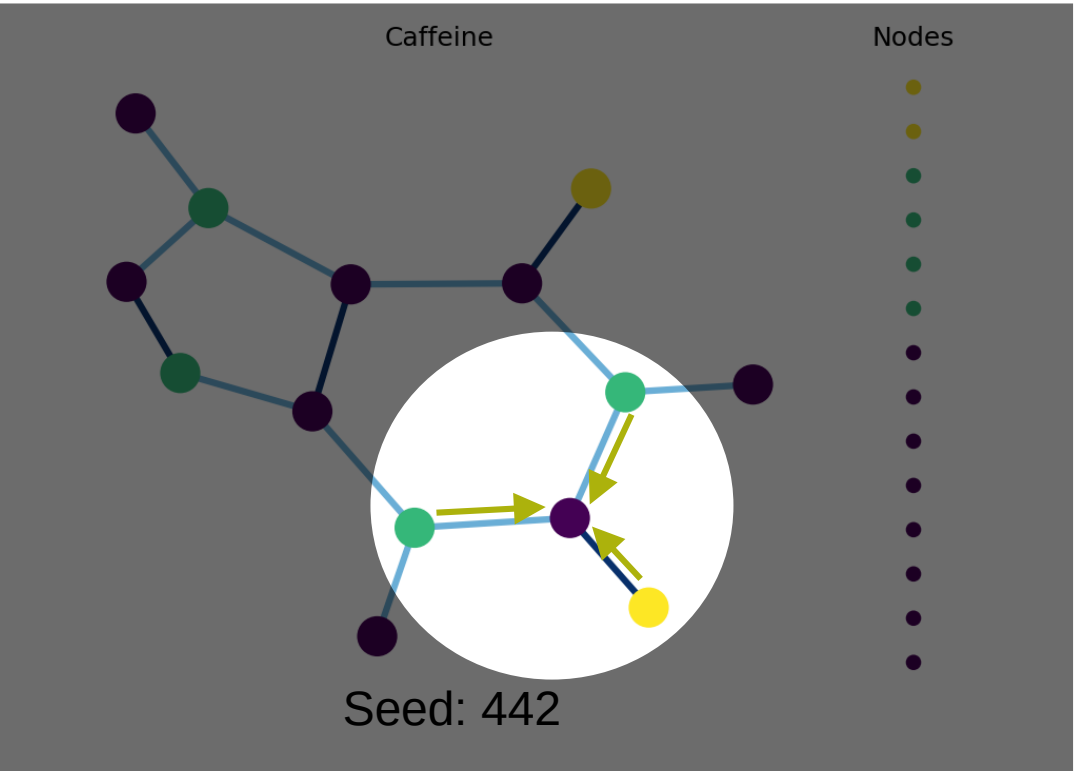
✓ Sum	-	Perm. Invariant
✓ Mean	-	Perm. Invariant
✓ Attention	-	Perm. Invariant
✗ Concat?	-	Perm. Sensitive

$$f(X) = f(PX)$$

$$f(X) = \sum_{\mathbf{x} \in X} \mathbf{x}$$

What happens locally in a GNN?

- Node neighborhoods remain the same under permutation



Parameterized Set Aggregation [1]:

Graph Neural Networks Backbone

$$f(\mathbf{v}_i) = \phi_\theta \left(\sum_{\mathbf{u} \in \mathcal{N}(\mathbf{v}_i)} \mathbf{u} \right)$$

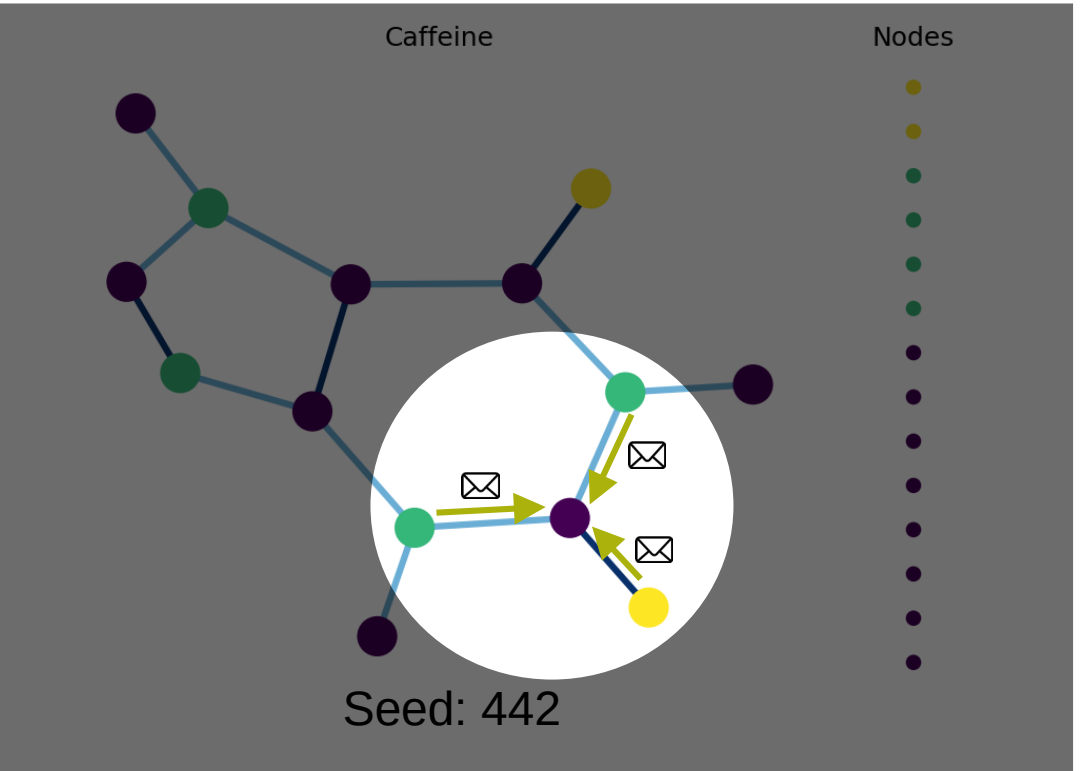
A function parameterized by theta, e.g. an MLP

Permutation invariant aggregation function (sum/mean/...)

[1] Zaheer et al. NeurIPS 2017 Deep Sets

Message Passing Principle

- Node neighborhoods remain the same under permutation



Parameterized Set Aggregation [1]:

Graph Neural Networks Backbone

$$f(\mathbf{v}_i) = \phi_\theta \left(\sum_{\mathbf{u} \in \mathcal{N}(\mathbf{v}_i)} \mathbf{u} \right)$$

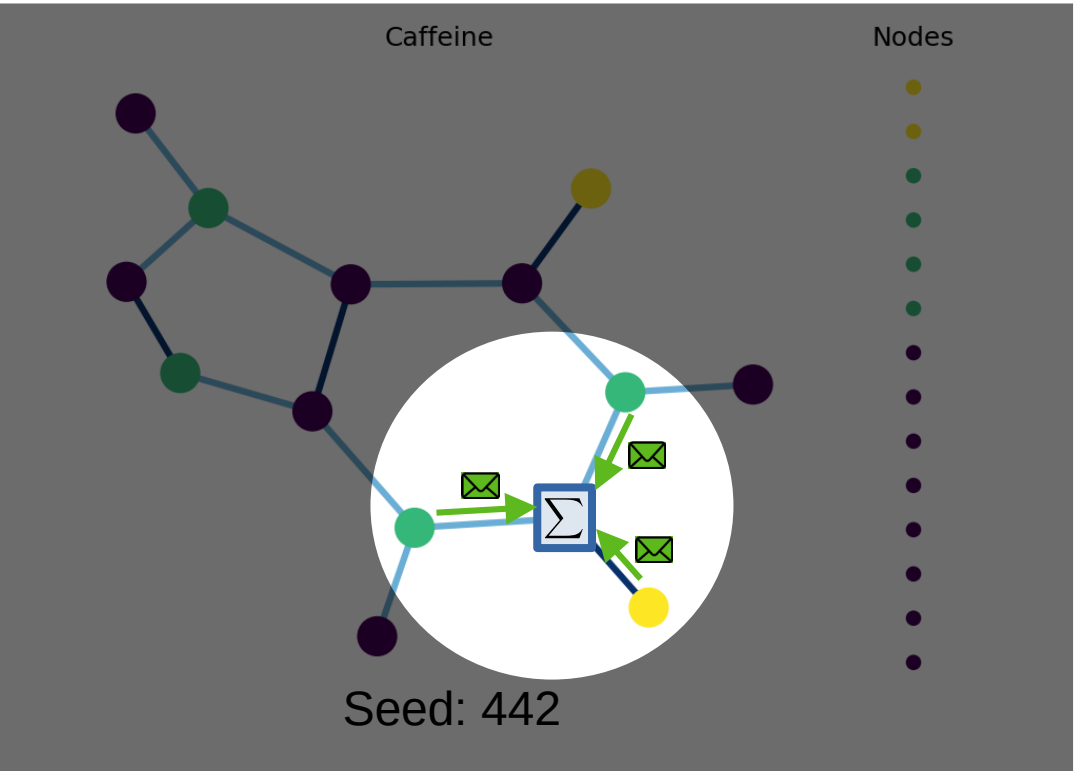
A function parameterized by theta, e.g. an MLP

Permutation invariant aggregation function (sum/mean/...)

[1] Zaheer et al. NeurIPS 2017 Deep Sets

Message Passing Principle

- Node neighborhoods remain the same under permutation



Parameterized Set Aggregation [1]:

Graph Neural Networks Backbone

1. Message Generation

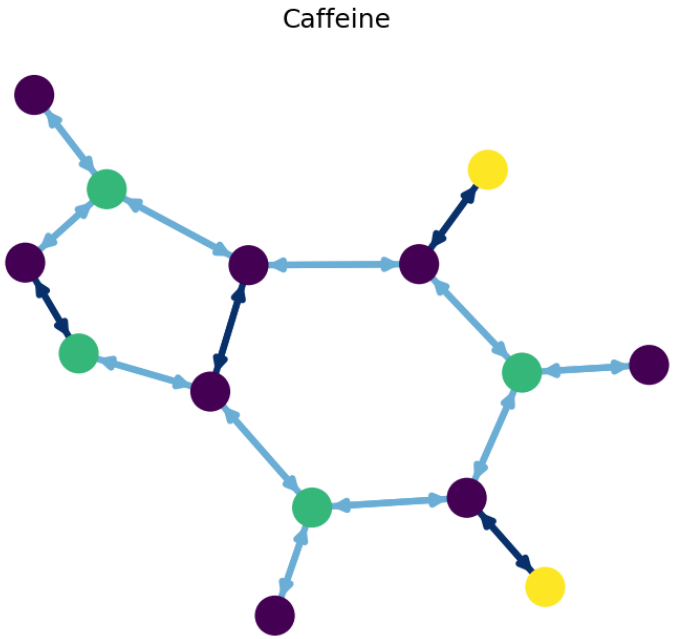
$$f(\mathbf{v}_i) = \phi_\theta \left(\sum_{\mathbf{u} \in \mathcal{N}(\mathbf{v}_i)} \psi(\mathbf{u}) \right)$$

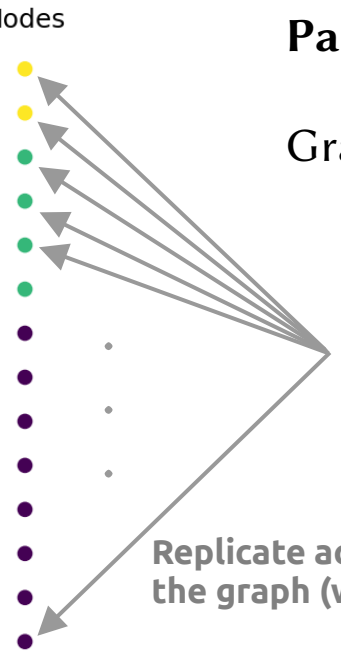
2. Message aggregation

[1] Zaheer et al. NeurIPS 2017 Deep Sets

Message Passing Principle

- Node neighborhoods remain the same under permutation





Parameterized Set Aggregation [1]:

Graph Neural Networks Backbone

1. Message Generation

$$f(\mathbf{v}_i) = \phi_\theta \left(\sum_{\mathbf{u} \in \mathcal{N}(\mathbf{v}_i)} \psi(\mathbf{u}) \right)$$

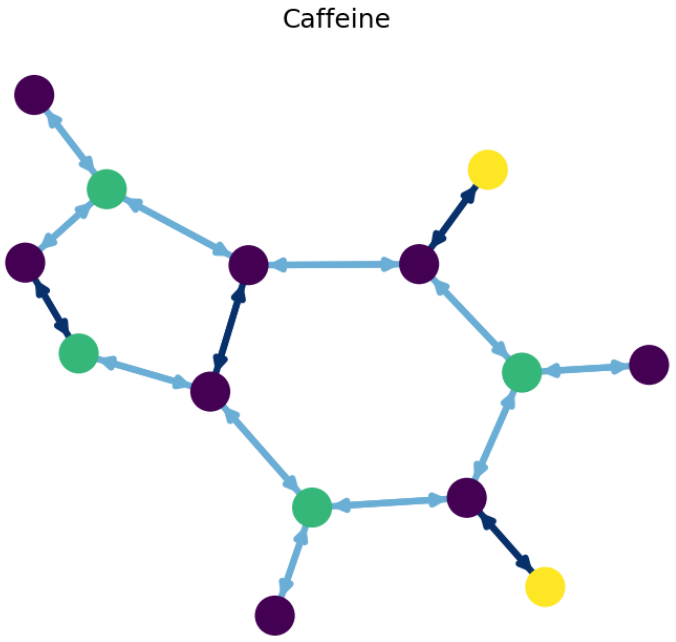
2. Message aggregation

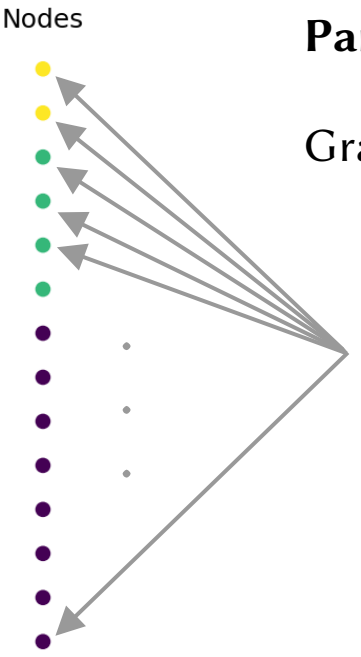
Replicate across the space of the graph (weights shared)

[1] Zaheer et al. NeurIPS 2017 Deep Sets

Message Passing Principle

- Node neighborhoods remain the same under permutation





Parameterized Set Aggregation [1]:

Graph Neural Networks Backbone

1. Message Generation

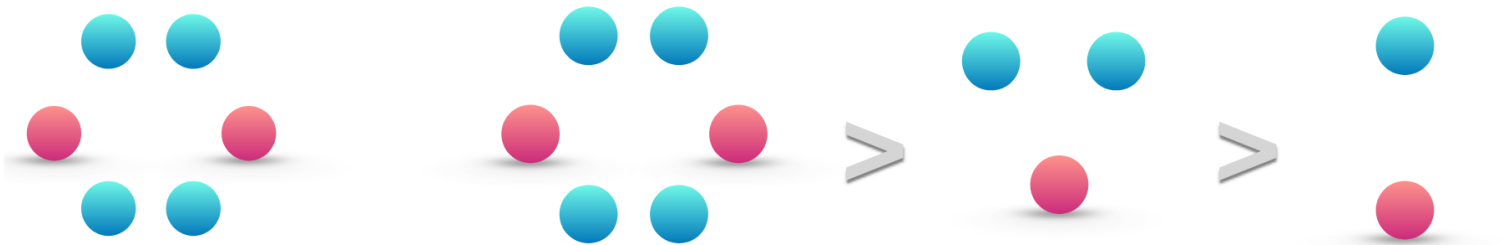
$$f(\mathbf{v}_i) = \phi_\theta \left(\sum_{\mathbf{u} \in \mathcal{N}(\mathbf{v}_i)} \psi(\mathbf{u}) \right)$$

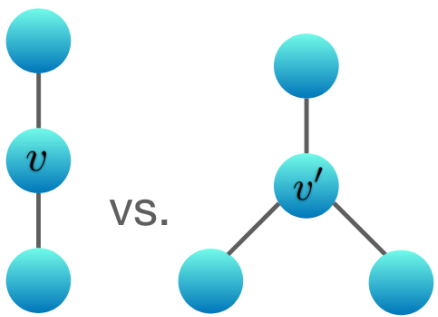
2. Message aggregation

The expressivity of a GNN is tied to the injectivity of this aggregation function.

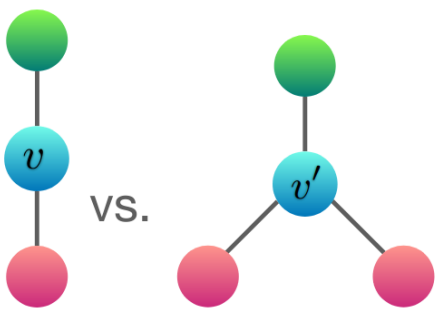
[1] Zaheer et al. NeurIPS 2017 Deep Sets

Expressivity of the Aggregation Function

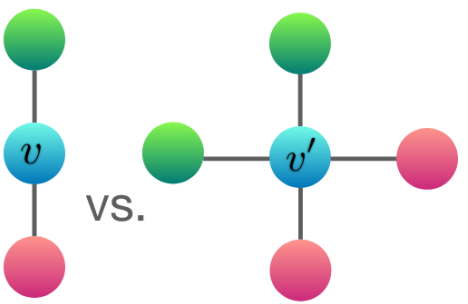




(a) Mean and Max both fail



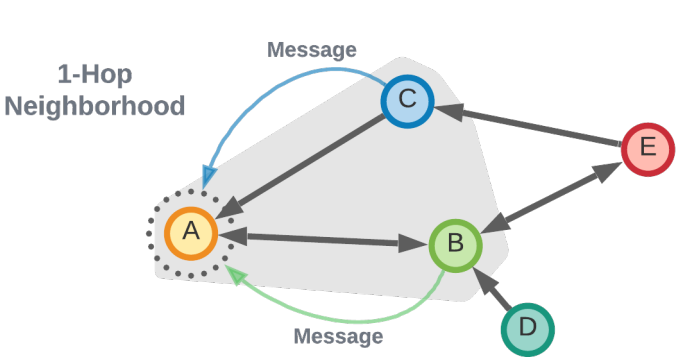
(b) Max fails

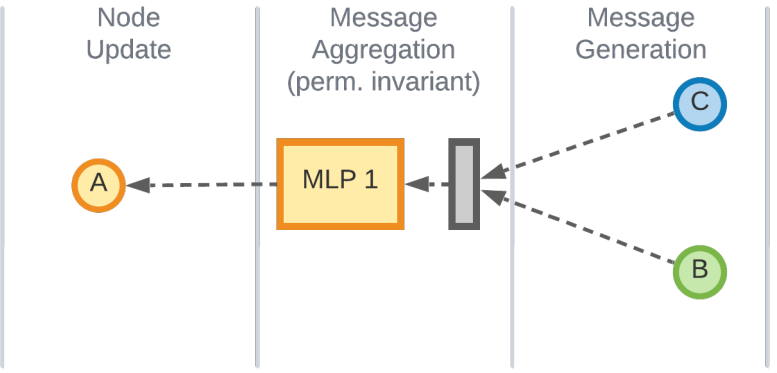


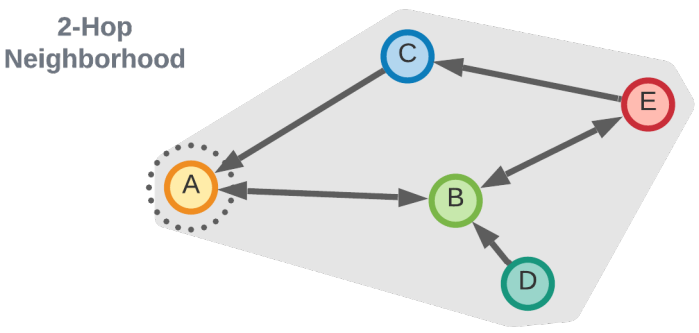
(c) Mean and Max both fail

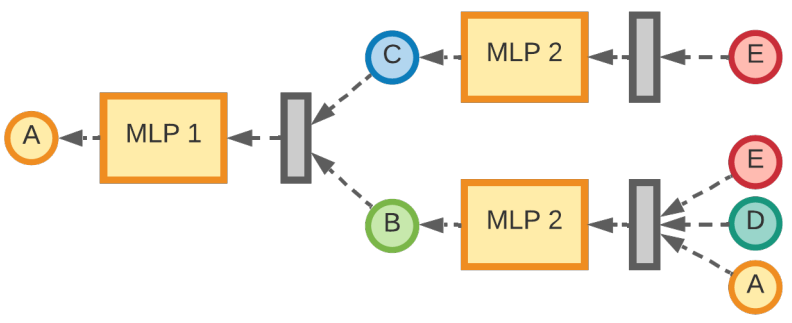
[1] Xu et al. (2018) How Powerful are Graph Neural Networks?

Message Passing Principle

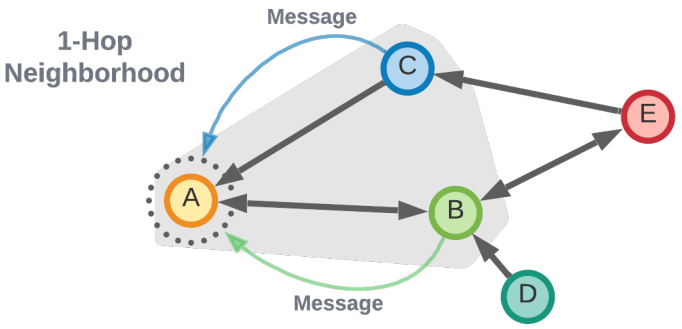


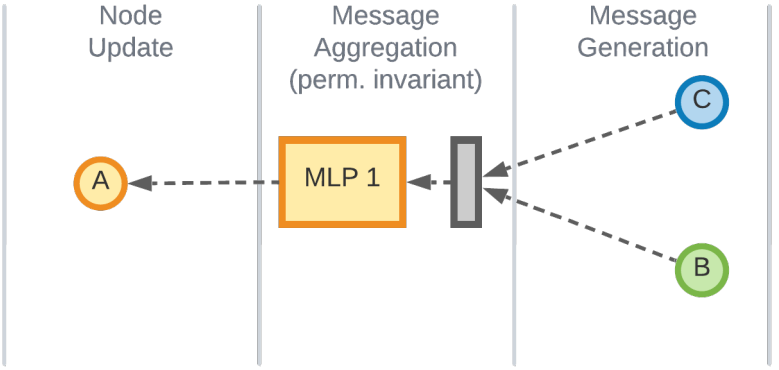


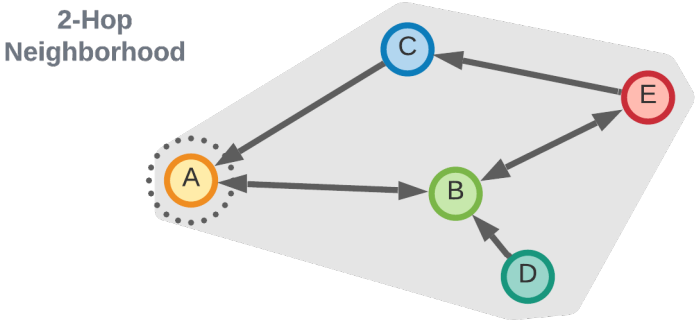




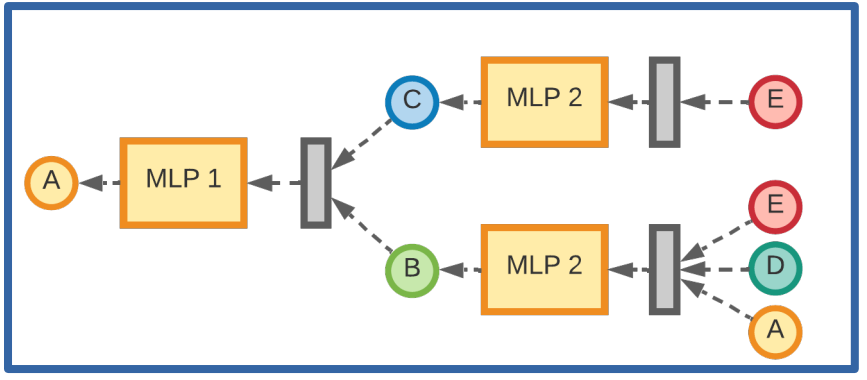
Message Passing Principle



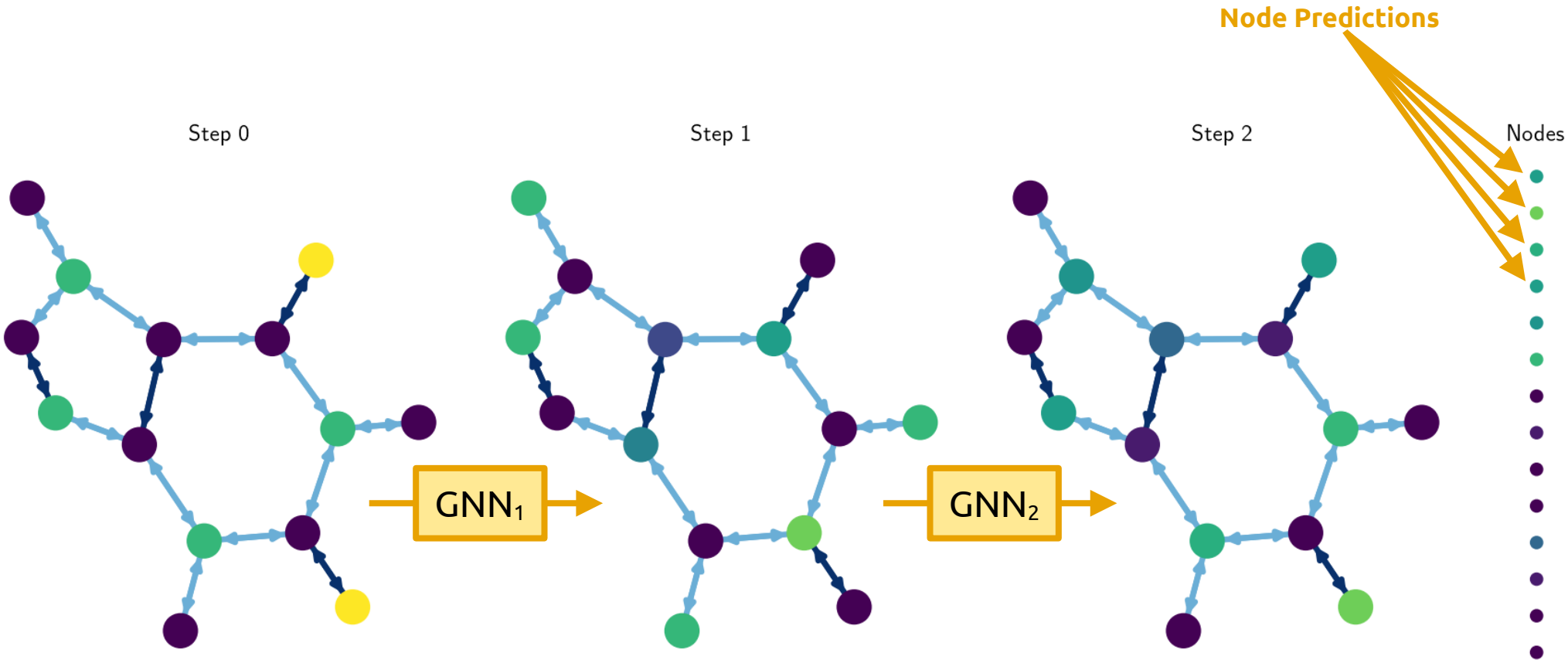




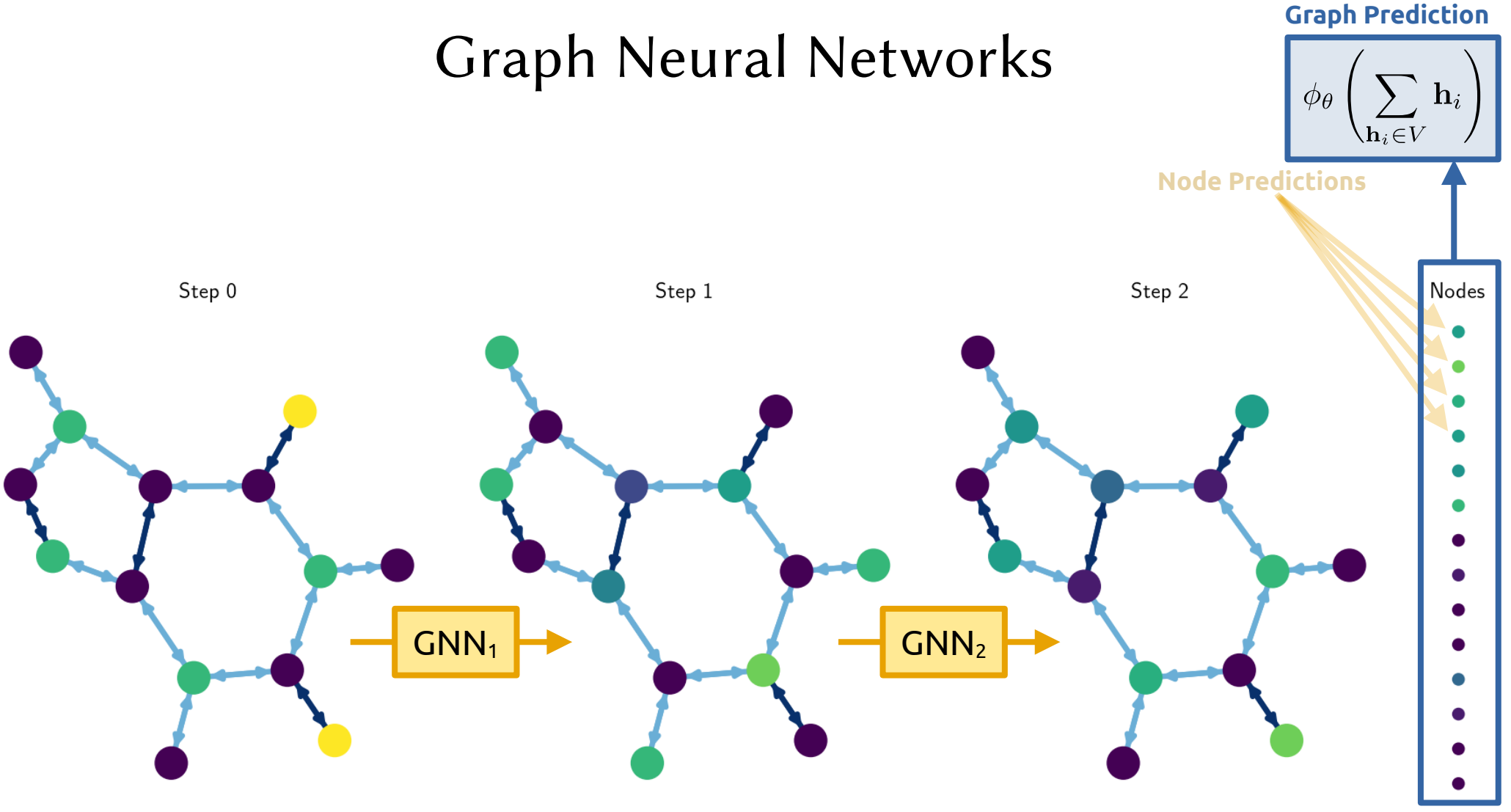
Computing Graph → Structural Sensitivity!



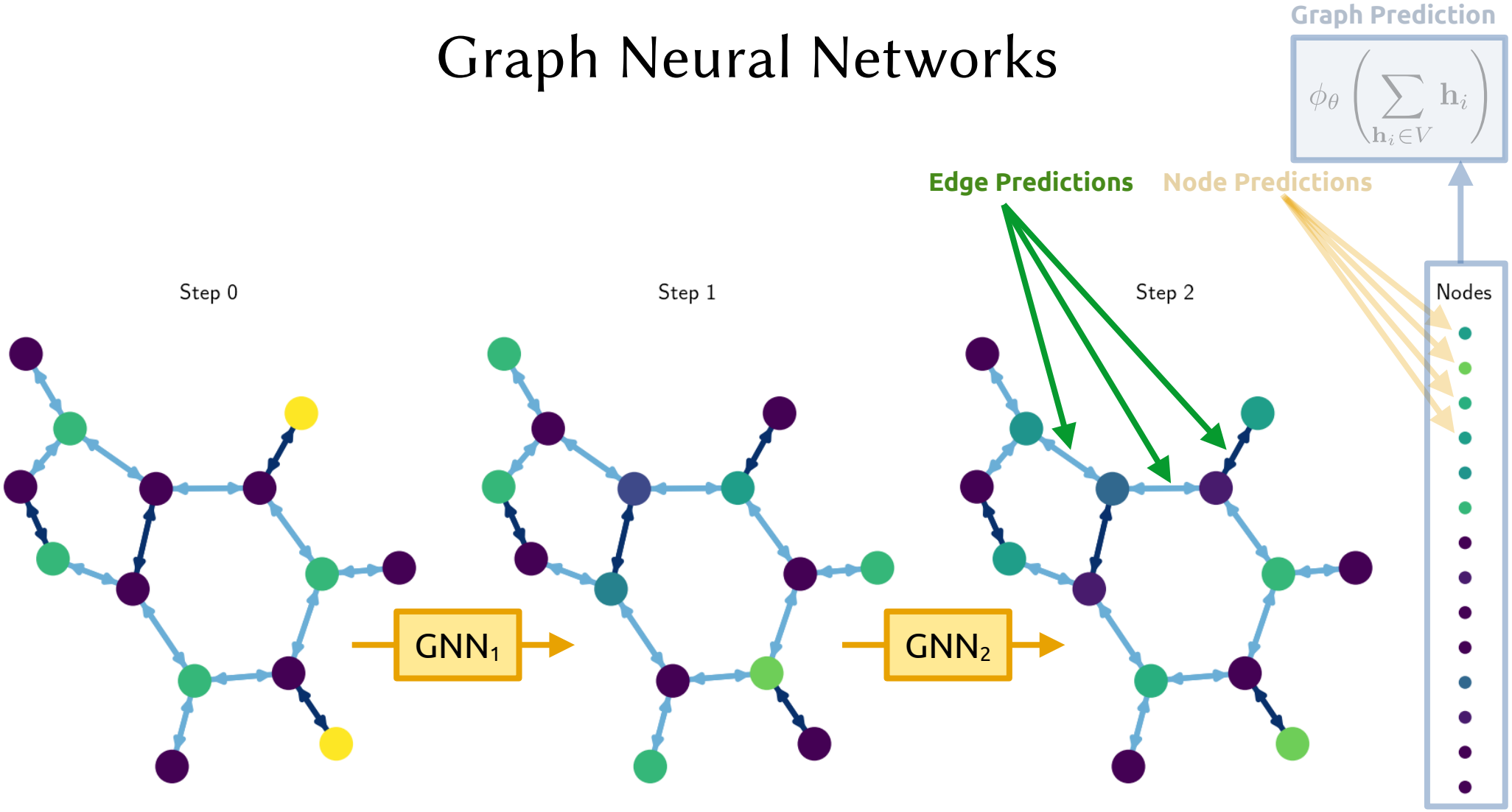
Graph Neural Networks



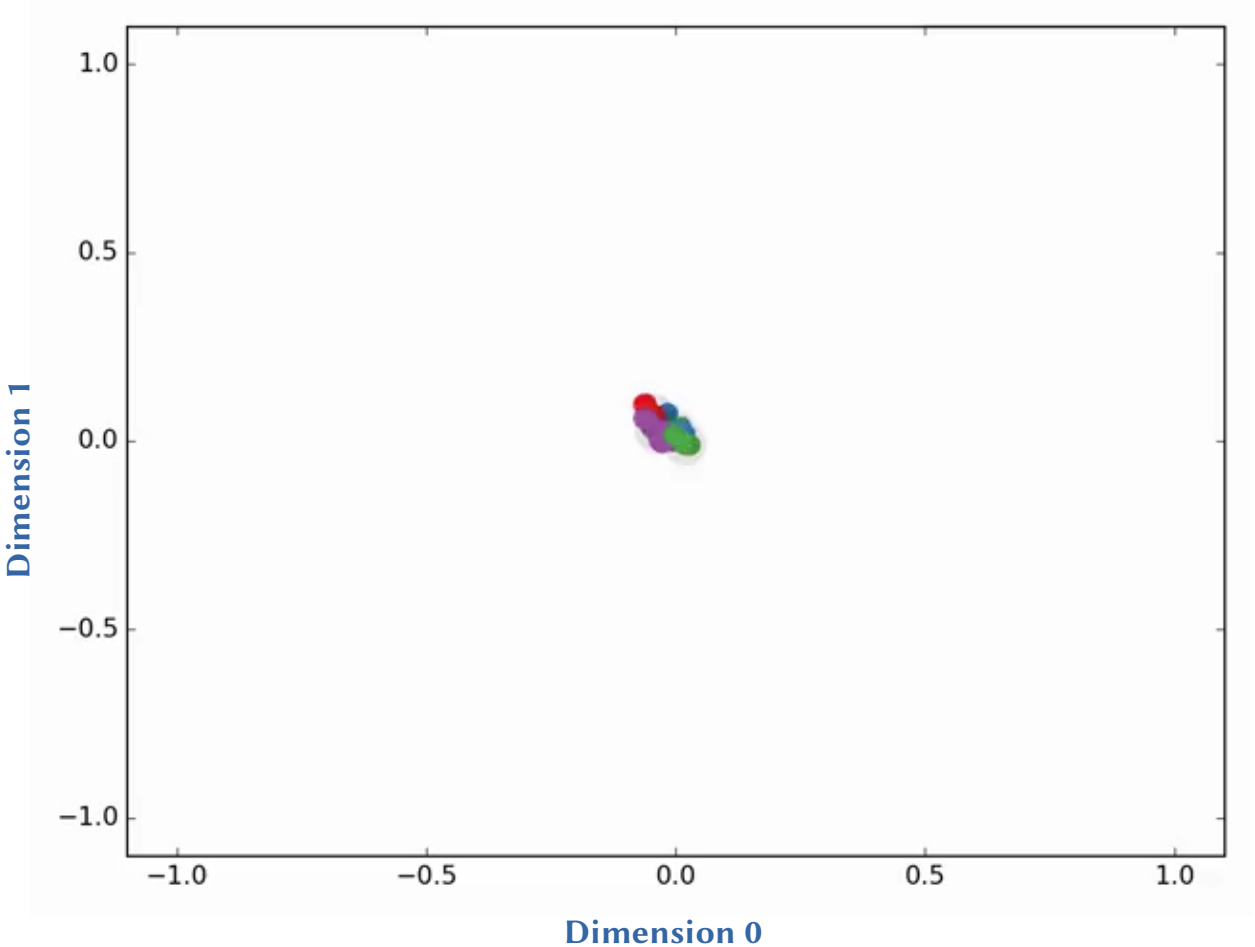
Graph Neural Networks

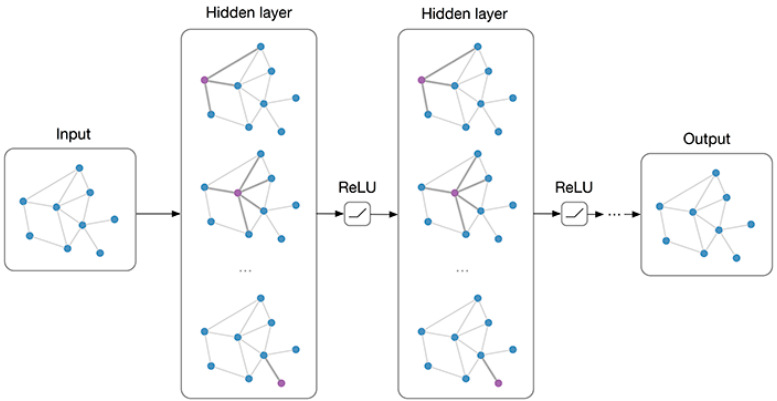


Graph Neural Networks



Latent Space of a GCN [1.1]





Model: GCN [1.2]
Time: Training Epochs
Embedding Size: 2

[1.1] Blog-Post (Video) https://tkipf.github.io/graph-convolutional-networks/
[1.2] Kipf et al. (2017) https://arxiv.org/abs/1609.02907

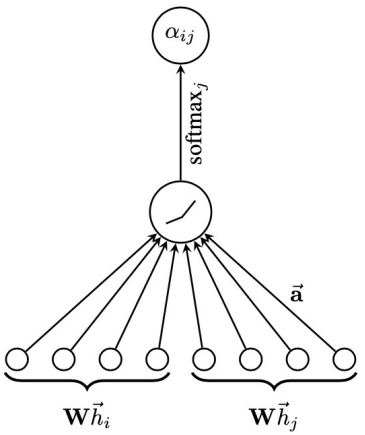
Example: Graph Attention Network [1]

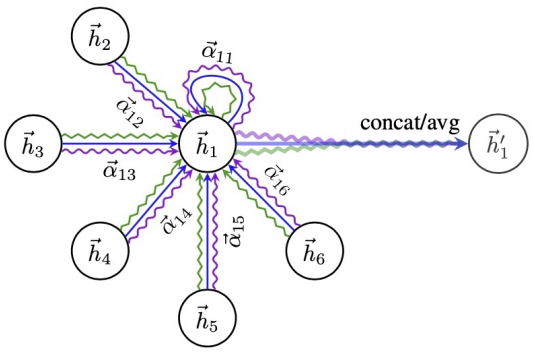
- This is the GAT - Graph Attention Network
- A parameterized attention function scales neighbors prior to aggregation
- This attending to node neighbors helps focusing on discriminative nodes

$$\mathbf{h}_v^k = \sigma_\theta \left(\sum_{u \in \mathcal{N}(v)} a_\psi(u, v) \mathbf{h}_u^{k-1} \right)$$

Attention Function weights each neighbor

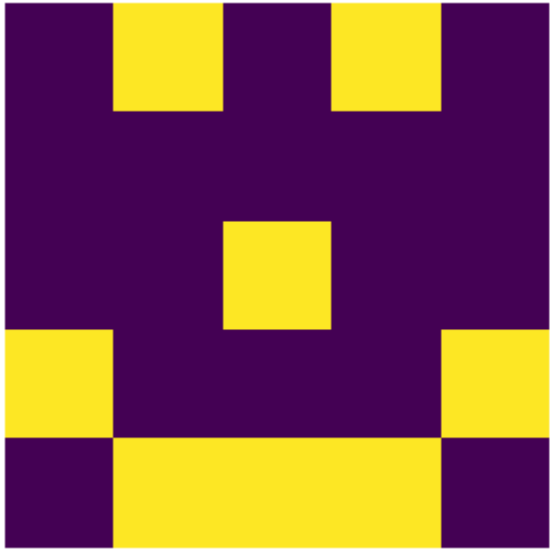
2. Message aggregation





[1] Veličković et al. (2017) Graph Attention Networks

Example: Images are Graphs



Image

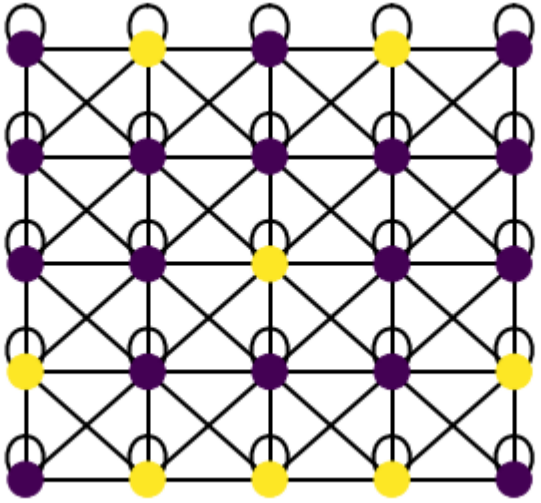
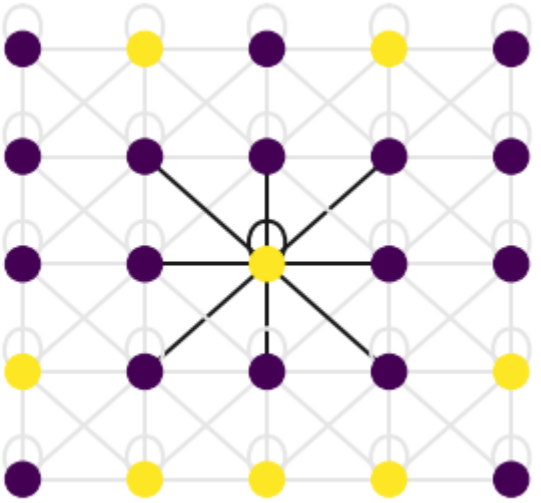
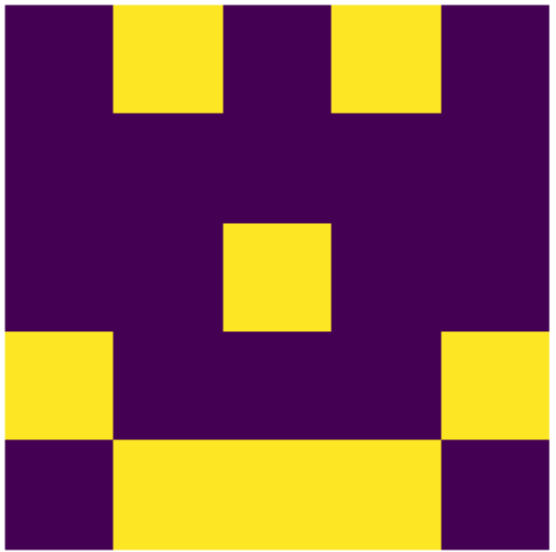


Image Graph

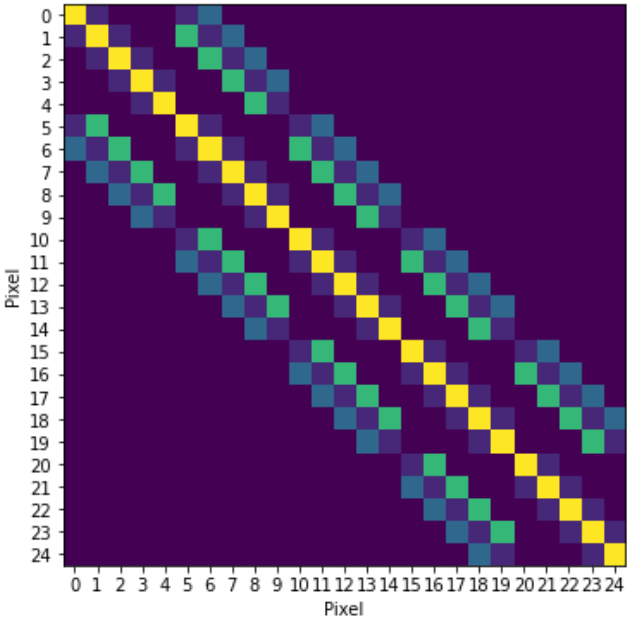


1-Hop Subgraph

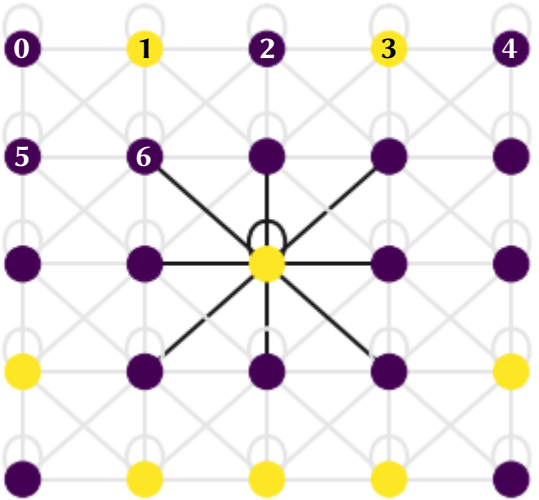
Example: Images are Graphs



Image

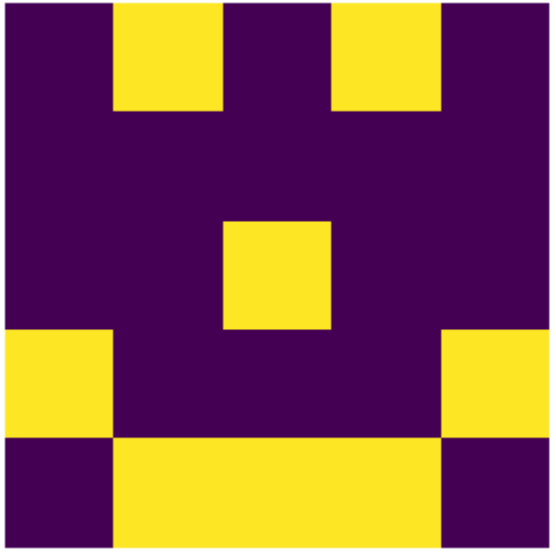


Adjacency Matrix

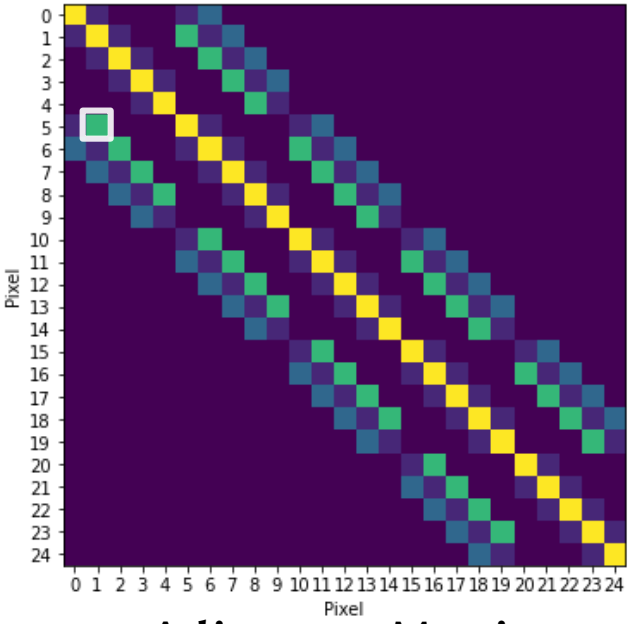


1-Hop Subgraph

Example: Images are Graphs

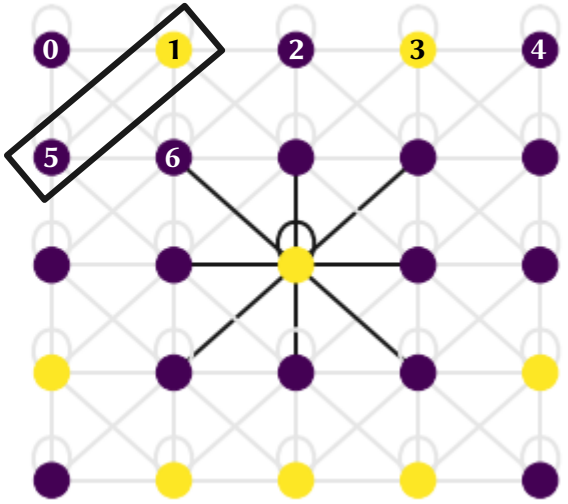


Image



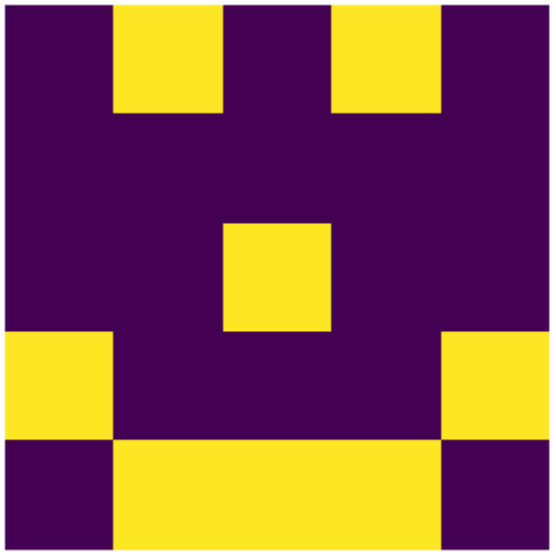
Adjacency Matrix

Pixel 5 is connected to Pixel 1

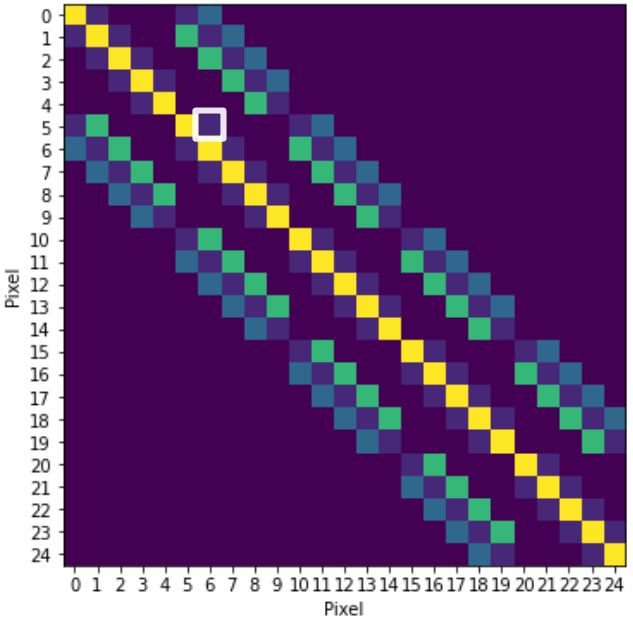


1-Hop Subgraph

Example: Images are Graphs

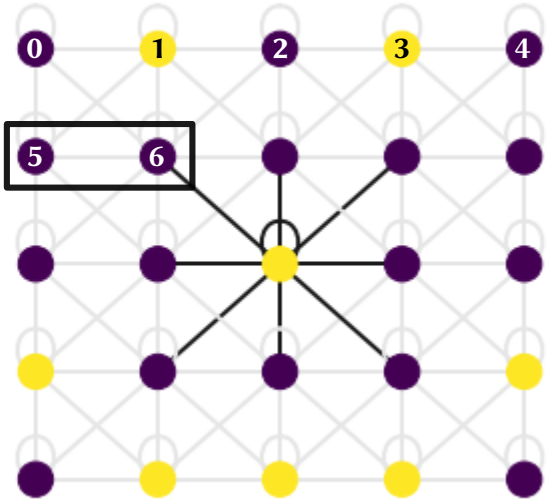


Image



Adjacency Matrix

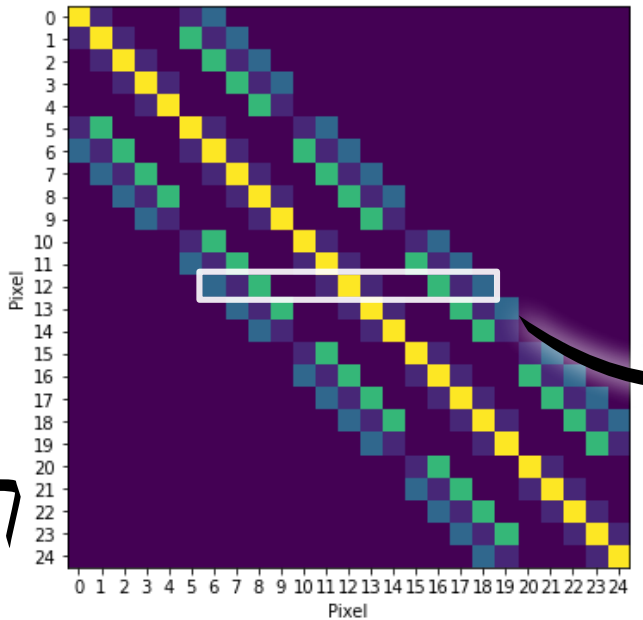
Pixel 5 is connected to Pixel 6



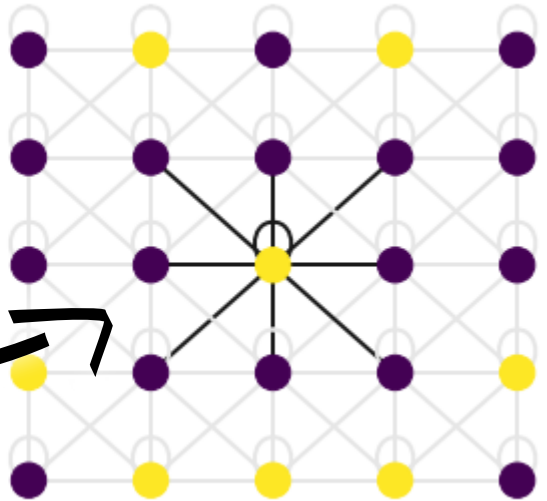
1-Hop Subgraph

Example: Images are Graphs

This this the circulant matrix used in ConvNets implementations, (different color denotes different kernel weight)



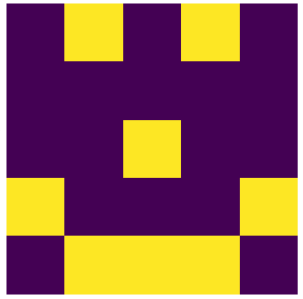
Adjacency Matrix



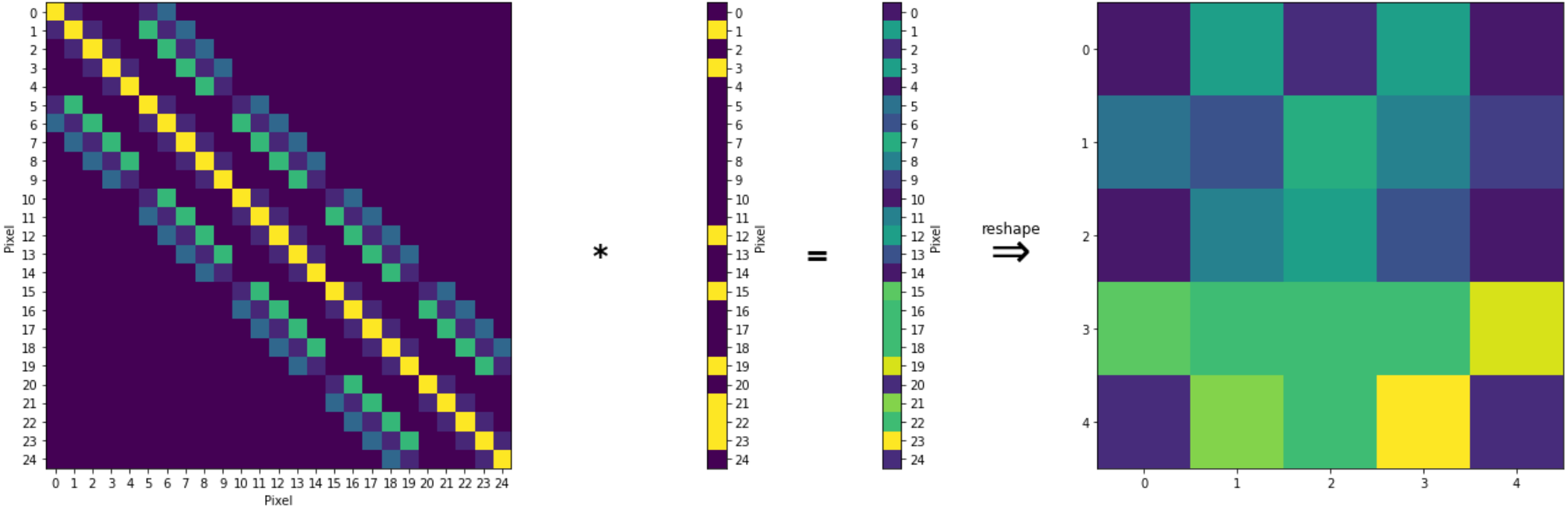
1-Hop Subgraph

Convolution Priors

- **Locality:**
  Applies a local kernel operation onto a neighborhood of pixels
- **Translational Symmetry:**
  A pattern is recognized independent of it's location
- **Cannonical Orientation:**
  Allows us to impose an ordering on the pixels



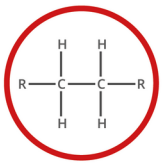
Cannonical Pixel Ordering



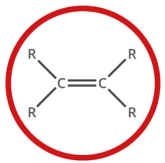
FUNCTIONAL GROUPS IN ORGANIC CHEMIST

FUNCTIONAL GROUPS ARE GROUPS OF ATOMS IN ORGANIC MOLECULES THAT ARE RESPONSIBLE FOR THE CHARACTERISTIC CHEMICAL REACTIONS OF THOSE
IN THE GENERAL FORMULAE BELOW, 'R' REPRESENTS A HYDROCARBON GROUP OR HYDROGEN, AND 'X' REPRESENTS ANY HALOGEN ATOM.

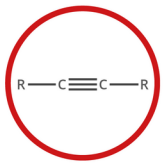
HYDROCARBONS | SIMPLE OXYGEN HETEROATOMICS | HALOGEN HETEROATOMICS | CARBONYL COMPOUNDS | NITROGEN BASED | SULFUR BASED



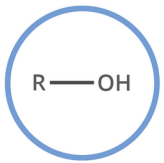
ALKANE
Naming: -ane
e.g. ethane



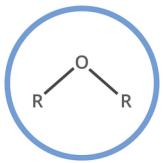
ALKENE
Naming: -ene
e.g. ethene



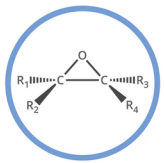
ALKYNE
Naming: -yne
e.g. ethyne



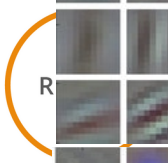
ALCOHOL
Naming: -ol
e.g. ethanol



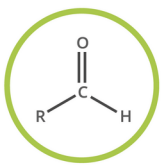
ETHER
Naming: -oxy -ane
e.g. methoxyethane



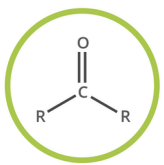
EPOXIDE
Naming: -ene oxide
e.g. ethene oxide



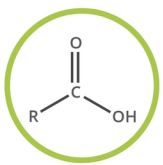
HAL
Nam
e.g. c



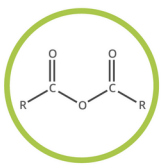
ALDEHYDE
Naming: -al
e.g. ethanal



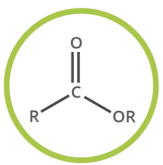
KETONE
Naming: -one
e.g. propanone



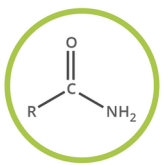
CARBOXYLIC ACID
Naming: -oic acid
e.g. ethanoic acid



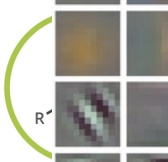
ACID ANHYDRIDE
Naming: -oic anhydride
e.g. ethanoic anhydride



ESTER
Naming: -yl -oate
e.g. ethyl ethanoate



AMIDE
Naming: -amide
e.g. ethanamide

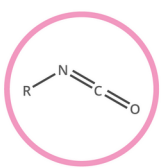


AC
Nami
e.g. eth

AMINE
Naming: -amine
e.g. ethanamine

NITRILE
Naming: -nitrile
e.g. ethanenitrile

IMINE
Naming: -imine
e.g. ethanimine

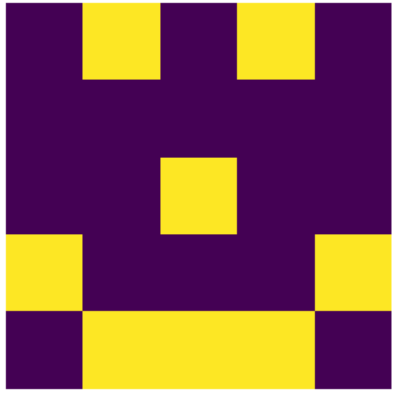


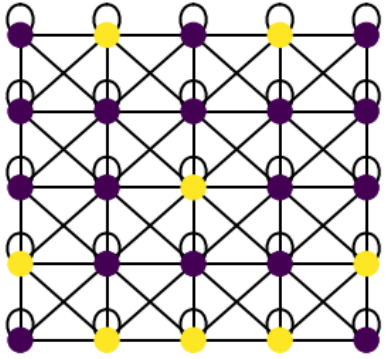
ISOCYANATE
Naming: -yl isocyanate
e.g. ethyl isocyanate

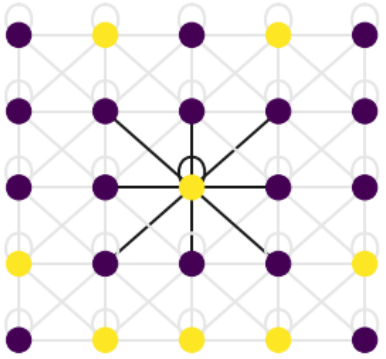
Local Motivs / Primitives that can be learned by **Local Functions**

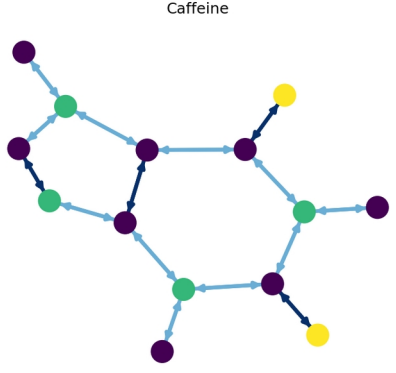
Image Source: https://chemistry.com.pk/infographics/functional-groups-in-organic-chemistry/

Questions so far?









1. Message Generation

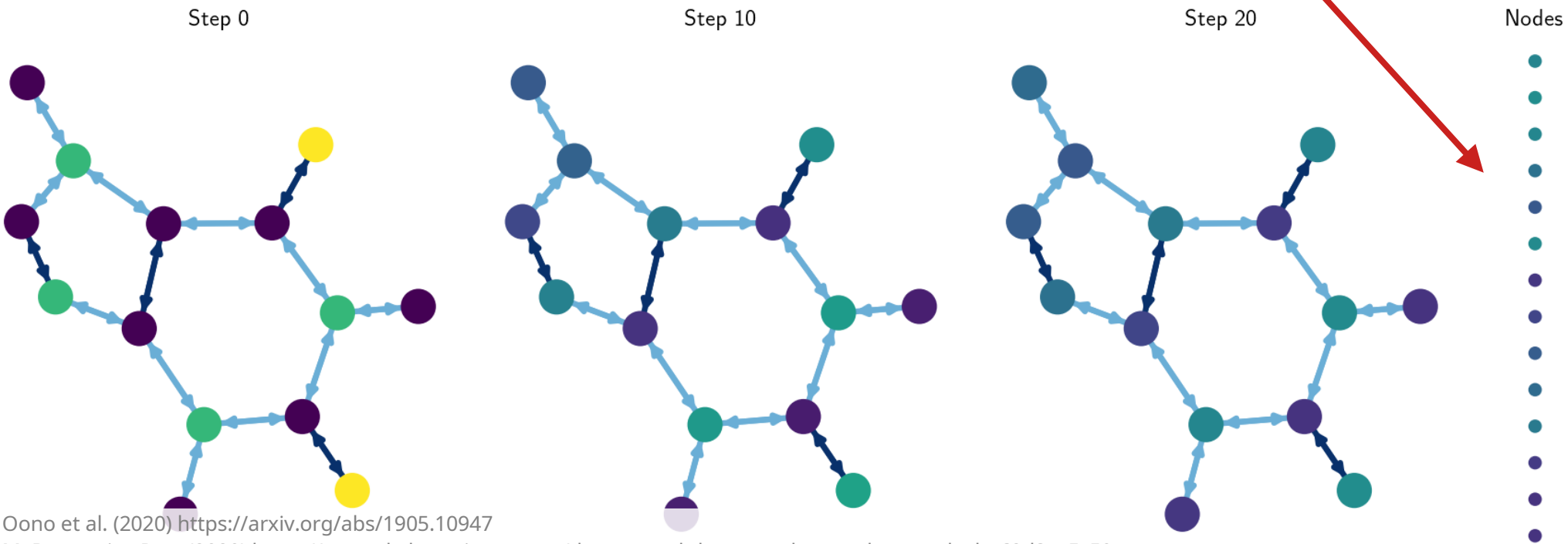
$$f(\mathbf{v}_i) = \phi_\theta \left(\sum_{\mathbf{u} \in \mathcal{N}(\mathbf{v}_i)} \psi(\mathbf{u}) \right)$$

2. Message aggregation

Deep Graph Neural Networks?

- Repetitive Aggregations are smoothing out the signal!
- This is the *Oversmoothing* Problem [1]
- Oversmoothing happens proportional to **graph diameter** and **node degree** [2]

The signal looses it's details!
Nodes become indistinguishable

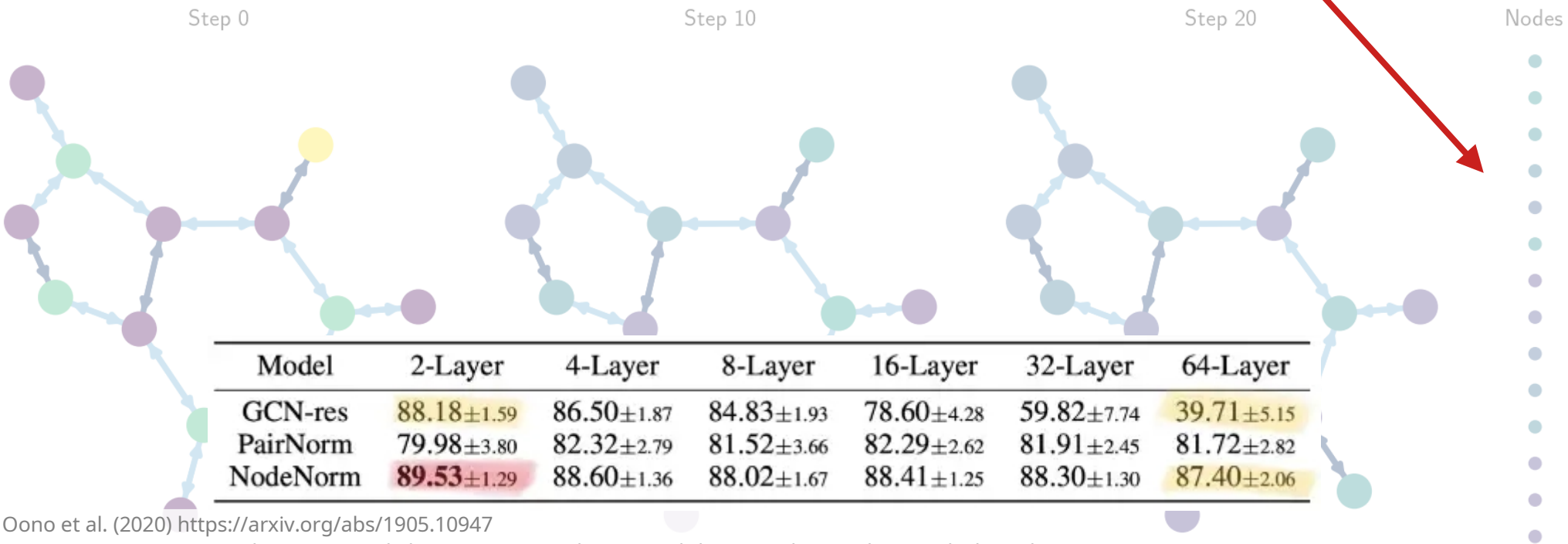


[1] Oono et al. (2020) https://arxiv.org/abs/1905.10947
[2] M. Bronsteins Post (2020) https://towardsdatascience.com/do-we-need-deep-graph-neural-networks-be62d3ec5c59

Deep Graph Neural Networks?

- Repetitive Aggregations are smoothing out the signal!
- This is the *Oversmoothing* Problem [1]
- Oversmoothing happens proportional to **graph diameter** and **node degree** [2]

The signal looses it's details!
Nodes become indistinguishable

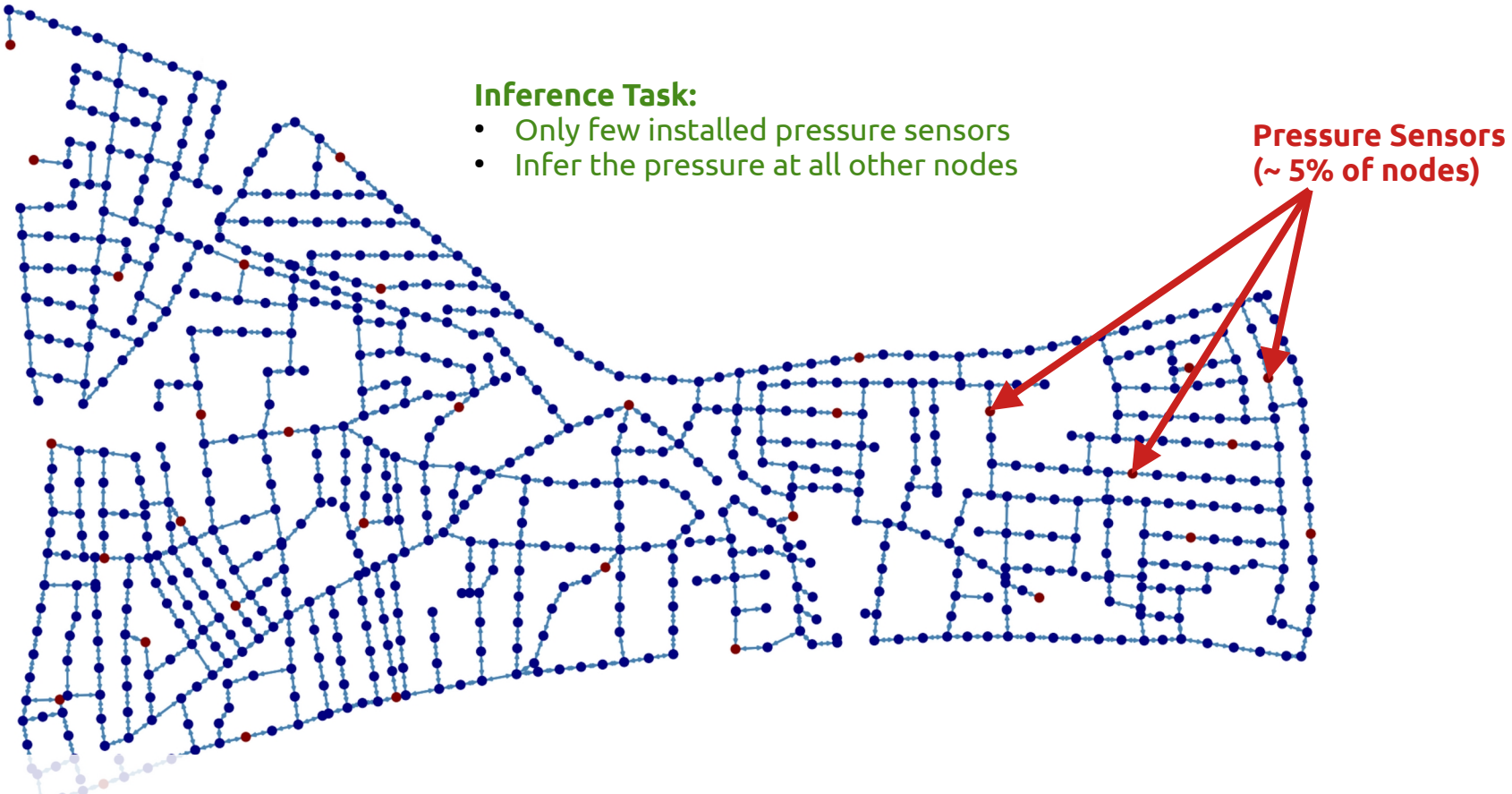


Model	2-Layer	4-Layer	8-Layer	16-Layer	32-Layer	64-Layer
GCN-res	88.18±1.59	86.50±1.87	84.83±1.93	78.60±4.28	59.82±7.74	39.71±5.15
PairNorm	79.98±3.80	82.32±2.79	81.52±3.66	82.29±2.62	81.91±2.45	81.72±2.82
NodeNorm	**89.53**±1.29	88.60±1.36	88.02±1.67	88.41±1.25	88.30±1.30	87.40±2.06

[1] Oono et al. (2020) https://arxiv.org/abs/1905.10947
[2] M. Bronsteins Post (2020) https://towardsdatascience.com/do-we-need-deep-graph-neural-networks-be62d3ec5c59

(~~Over-~~)Smoothing is good sometimes!

Case-Study: Water Distribution Systems (WDS) [1]



[1] I. Ashraf, L. Hermes, B. Hammer (2022) Spatial Graph Convolution Neural Networks for Water Distribution Systems; https://arxiv.org/abs/2211.09587

(~~Over-~~)Smoothing is good sometimes!

Case-Study: Water Distribution Systems (WDS) [1]

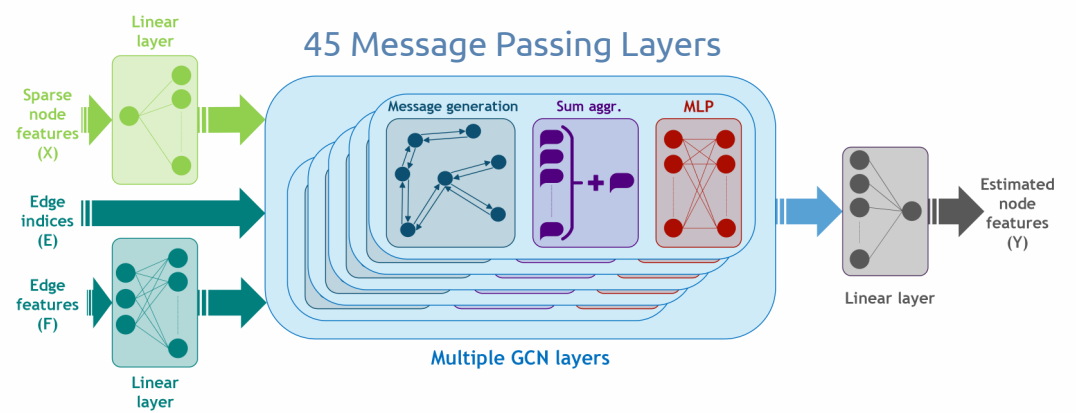
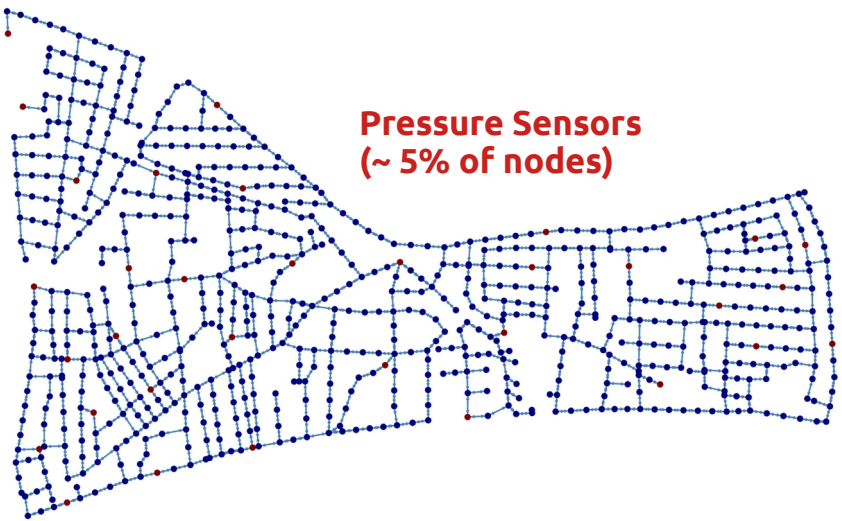


Table 4. Mean errors across nodes and samples on L-Town.

	Model	Error ($\times 10^{-3}$)		
		All	Sensor	Non-sensor
		Smooth Data		
baseline	ChebNet	2.55 ± 2.87	2.38 ± 3.55	2.55 ± 2.83
ours	m-GCN (45 x 1)	**0.39** ± 0.37	**0.43** ± 0.52	**0.39** ± 0.36
ours	m-GCN (10 x 5)	0.83 ± 0.68	0.74 ± 0.59	0.83 ± 0.69
		Noisy Data		
baseline	ChebNet	2.92 ± 3.35	2.78 ± 4.02	2.93 ± 3.32
ours	m-GCN (45 x 1)	**0.54** ± 0.75	**0.64** ± 1.06	**0.53** ± 0.73
ours	m-GCN (10 x 5)	0.90 ± 0.82	0.81 ± 0.74	0.90 ± 0.83

Inference Task:

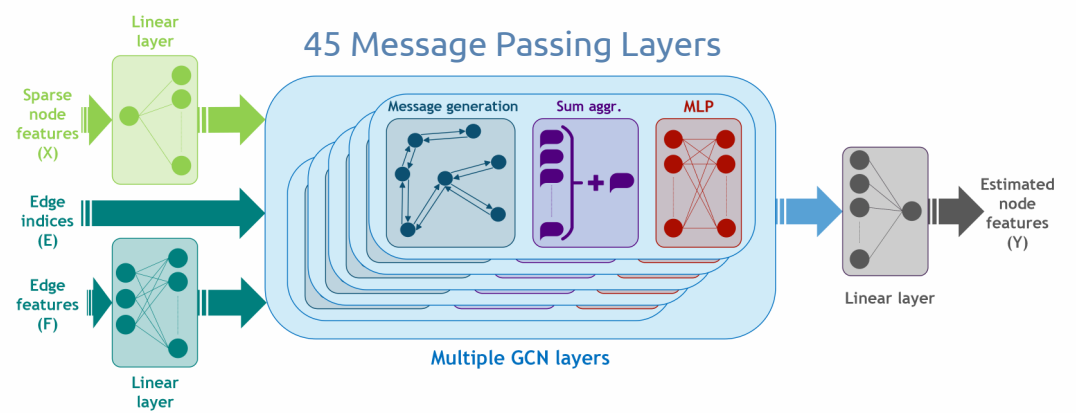
- Only few installed pressure sensors
- Infer the pressure at all other nodes



[1] I. Ashraf, L. Hermes, B. Hammer (2022) Spatial Graph Convolution Neural Networks for Water Distribution Systems; https://arxiv.org/abs/2211.09587

(~~Over-~~)Smoothing is good sometimes!

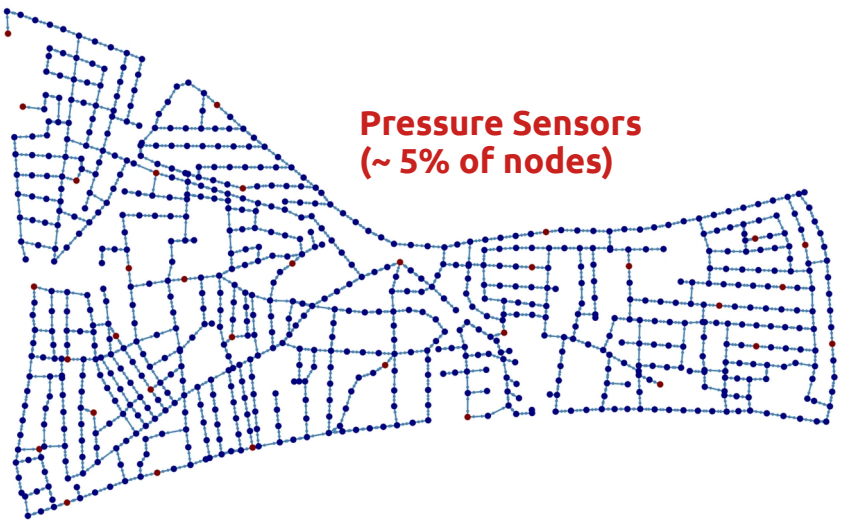
Case-Study: Water Distribution Systems (WDS) [1]



- **Architecture**: 45 Layers - Very deep GNN
- **Empirically**: Less Layers results in Performance drop
- **Intuition**: Water in a WDS smoothes out perturbations over the space of the graph ⟶ GCN-smoothing might be beneficial here.

Inference Task:

- Only few installed pressure sensors
- Infer the pressure at all other nodes



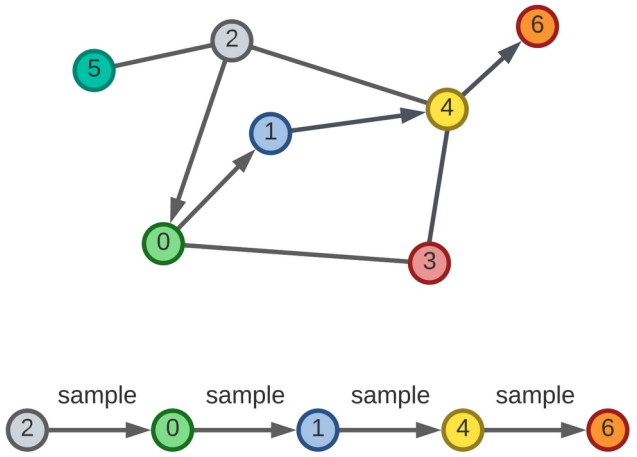
[1] I. Ashraf, L. Hermes, B. Hammer (2022) Spatial Graph Convolution Neural Networks for Water Distribution Systems; https://arxiv.org/abs/2211.09587

Beyond Aggregation-Based Methods

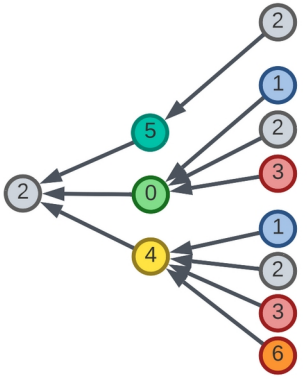
- In other applications **oversmoothing** can prevent learning
- Real-world graphs can be noisy, aggregating **noise can prevent learning**
- **Information** probably **not uniformly distributed** on a graph
- Graph sampling can **focus computational resources** to specific subgraphs
- My current project: Sampling-based GNNs!

Sampling-Based GNN

- Idea: Instead of aggregating neighborhoods, **sample the neighborhood intelligently**



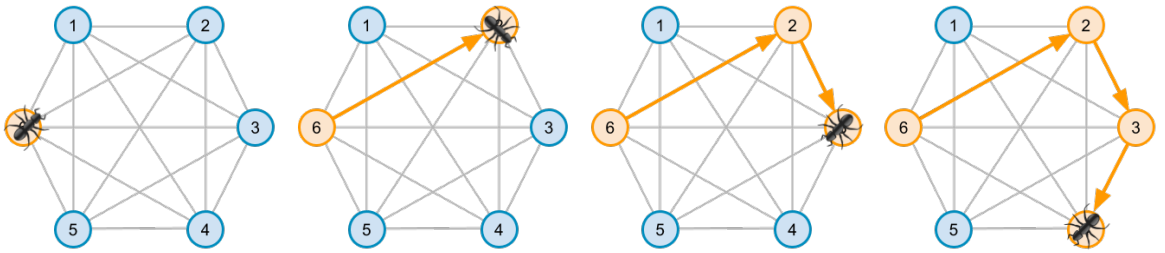
Sampling-Based GNN



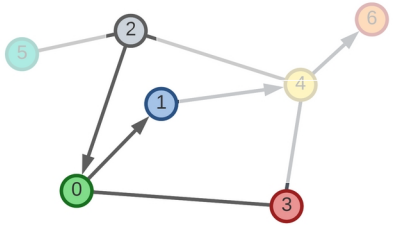
GCN

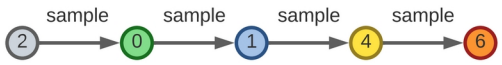
- Motivation:
  - Aggregation helps with encoding structure, but causes problems
  - Sampling individual neighbors reduces over-squashing and over-smoothing
  - Information on a graph is not necessarily dense, but may be sparse (e.g. Molecules)

Conceptualizing the Idea

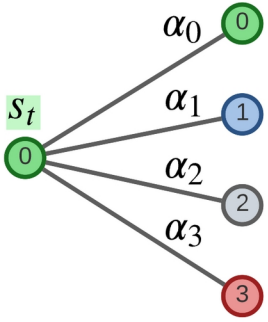


- Differentiable Exploration of Graphs by Independent '*Samplers', here Ants*
- This allows multiple extensions that GCNs cannot apply
  - Communication between samplers visiting the same node (Doubles the receptive field)
  - Update of nodes on the sampling trajectory
  - Out-of-the-box explainability by observing information flow (?)





$s_t = [1, 0, 0, .., 0]$



$\alpha = [-0.5, 0.2, -0.8, 0.4]$

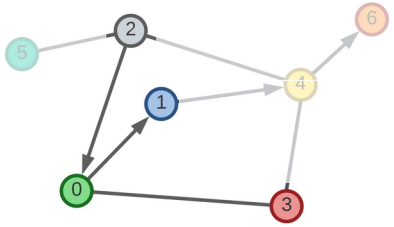
1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u|h_v)$$

Sampling-Based GNN

- We can use neighborhood attention just like a GAT to *score* nodes
  - For each node $u \in \mathcal{N}_v$

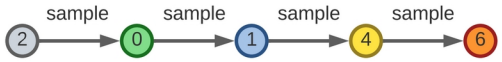
$$\alpha_v = MLP(h_v|h_u)$$



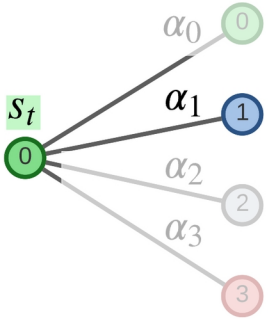
Sampling-Based GNN

- We can use neighborhood attention to *score* nodes
  - For each node $u \in \mathcal{N}_v$

$$\alpha_v = MLP(h_v|h_u)$$



$s_t = [1, 0, 0, \ldots, 0]$

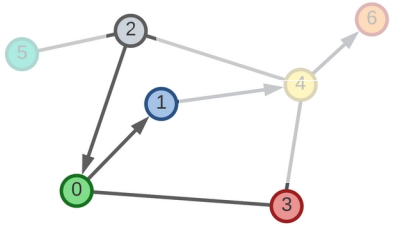


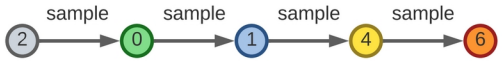
Selecting the highest alpha, would yield a gradient for ONLY that neighbor.

$\alpha = [-0.5, 0.2, -0.8, 0.4]$

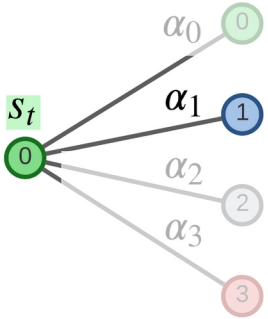
1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u|h_v)$$





$s_t = [1, 0, 0, \ldots, 0]$



$\alpha = [-0.5, 0.2, -0.8, 0.4]$

1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u|h_v)$$

Sampling-Based GNN

- We can use neighborhood attention to *score* nodes
  - For each node $u \in \mathcal{N}_v$

$$\alpha_v = MLP(h_v|h_u)$$

- We can use a relaxation of argmax: Softmax with temperature

$$p_t = \text{softmax}\left(\frac{\vec{\alpha}_{t,e_v}}{\tau}\right)$$

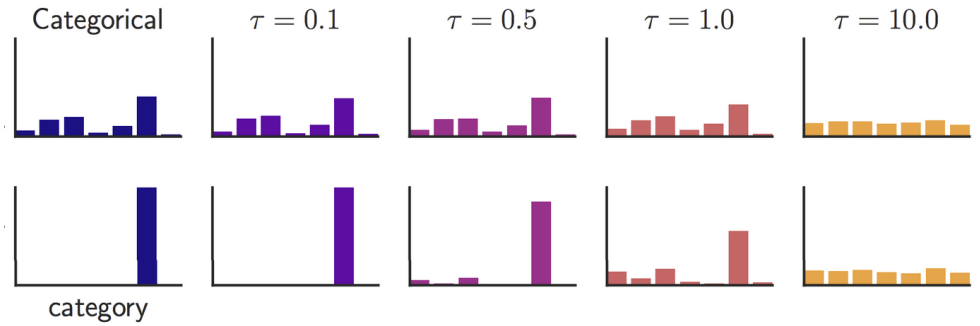
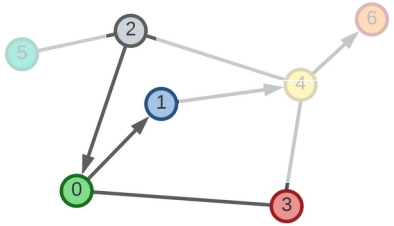
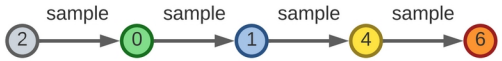
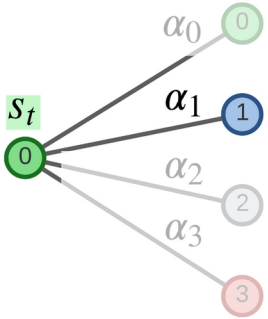


Figure From: https://fabianfuchsml.github.io/gumbel/





$s_t = [1, 0, 0, \ldots, 0]$



$\alpha = [-0.5, 0.2, -0.8, 0.4]$

1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u|h_v)$$

Sampling-Based GNN

- We can use neighborhood attention to *score* nodes
  - For each node $u \in \mathcal{N}_v$

$$\alpha_v = MLP(h_v|h_u)$$

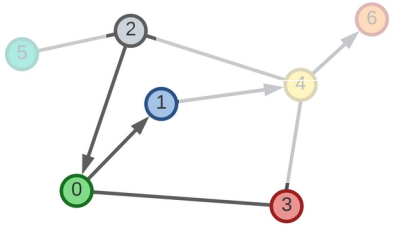
- We can use a relaxation of argmax: Softmax with temperature

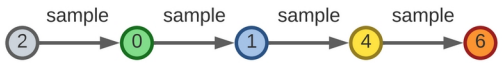
$$p_t = \text{softmax}\left(\frac{\vec{\alpha}_{t,e_v}}{\tau}\right)$$

- This is the still a relaxation, but we want only a single node to be sampled from this distribution

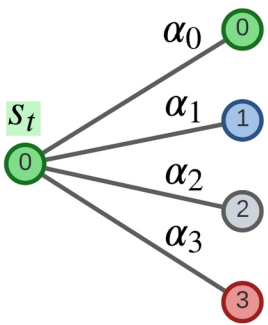
Figure From: https://fabianfuchsml.github.io/gumbel/

Straight-Through Gumbel-Softmax





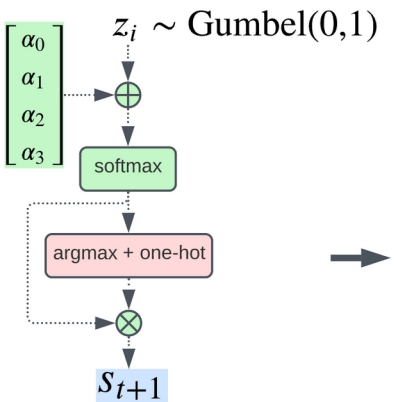
$s_t = [1, 0, 0, .., 0]$



$\alpha = [-0.5, 0.2, -0.8, 0.4]$

1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u|h_v)$$



$s_{t+1} = [0, 1, 0, .., 0]$

2. Sample Gumbel softmax to find the next neighbor (non-deterministic)

3. Result: winner-takes-all neighborhood sample.

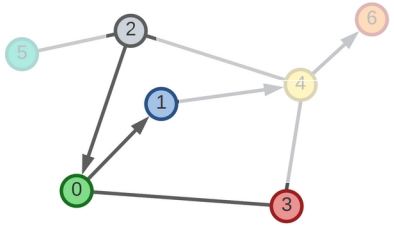
- The Gumbel-Softmax is a reparameterizable categorical probability function
- We use the Gumbel-Softmax Trick to sample one node from the edge distribution

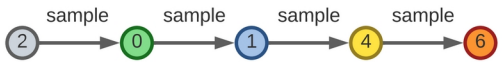
$$\mathbf{p}_t = \text{softmax}\left(\frac{z_{t,e_v} + \vec{\alpha}_{t,e_v}}{\tau}\right)$$

$$\mathbf{s}^{t+1} = \text{one-hot}\left(\text{argmax}_{\forall e_v \in \mathcal{N}}\ \mathbf{p}_t\right)$$

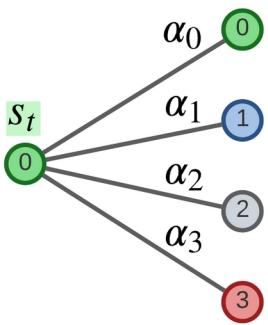
[1] Bengio et al.: Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation

Straight-Through Gumbel-Softmax





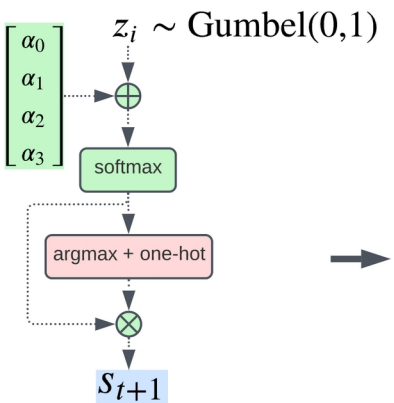
$s_t = [1, 0, 0, .., 0]$



$\alpha = [-0.5, 0.2, -0.8, 0.4]$

1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u|h_v)$$



$s_{t+1} = [0, 1, 0, .., 0]$

2. Sample Gumbel softmax to find the next neighbor (non-deterministic)

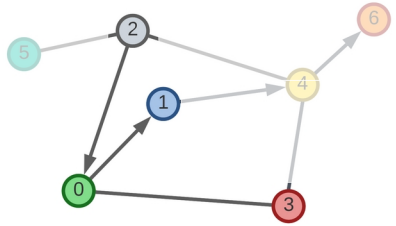
3. Result: winner-takes-all neighborhood sample.

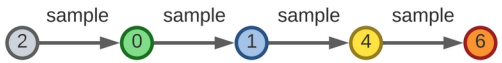
- This way we can generate walks along the graph that are trainable
- Integrating the walk with a sequential model yields the node embedding

$$\mathbf{p}_t = \text{softmax}\left(\frac{z_{t,e_v} + \vec{\alpha}_{t,e_v}}{\tau}\right)$$
$$\mathbf{s}^{t+1} = \text{one-hot}\left(\text{argmax}_{\forall e_v \in \mathcal{N}}\ \mathbf{p}_t\right)$$
$$\mathbf{h}'_v = \sum_{v \in \mathcal{N}(u)} s_v^{t+1} \mathbf{h}_v$$
$$\bar{\mathbf{s}}^{t+1} = \text{MLP}(\bar{\mathbf{s}}^t + \mathbf{h}'_v)$$

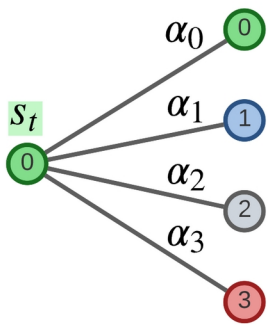
[1] Bengio et al.: Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation

Straight-Through Gumbel-Softmax





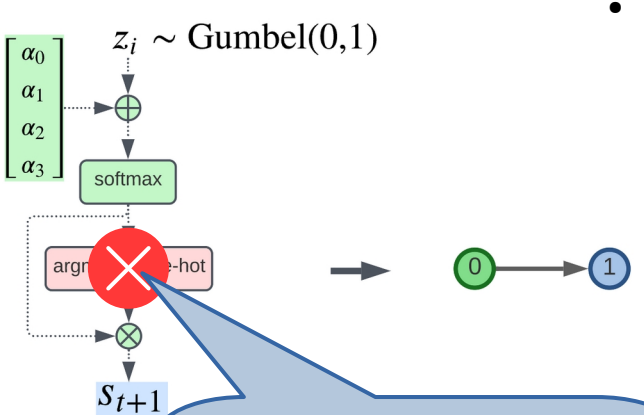
$s_t = [1, 0, 0, \ldots, 0]$



$\alpha = [-0.5, 0.2, -0.8, 0.4]$

1. Compute edge logits of neighborhood (similar to a GAT)

$$\alpha_v = MLP(h_u | h_v)$$



$s_{t+1} = [0,$

2. Sample Gu
to find the ne
(non-determin
-all

Argmax is non-differentiable, in the backward pass, we simply bypass it and use a *biased gradient*

- This way we can generate walks along the graph that are trainable
- Integrating the walk with a sequential model yields the node embedding

$$\mathbf{p}_t = \text{softmax}\left(\frac{z_{t,e_v} + \vec{\alpha}_{t,e_v}}{\tau}\right)$$

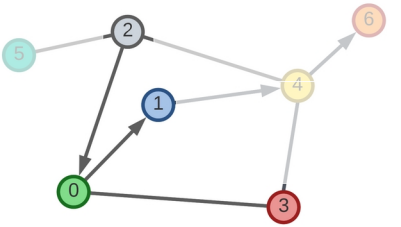
$$\mathbf{s}^{t+1} = \text{one-hot}\left(\text{argmax}_{\forall e_v \in \mathcal{N}}\ \mathbf{p}_t\right)$$

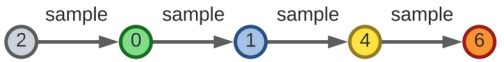
$$\mathbf{h}'_v = \sum_{v \in \mathcal{N}(u)} s_v^{t+1} \mathbf{h}_v$$

$$\bar{\mathbf{s}}^{t+1} = \text{MLP}(\bar{\mathbf{s}}^t + \mathbf{h}'_v)$$

[1] Bengio et al.: Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation

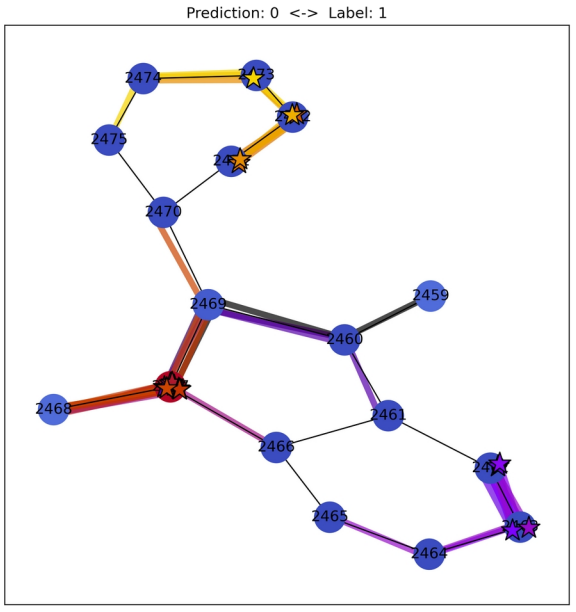
Sampling-Based GNN

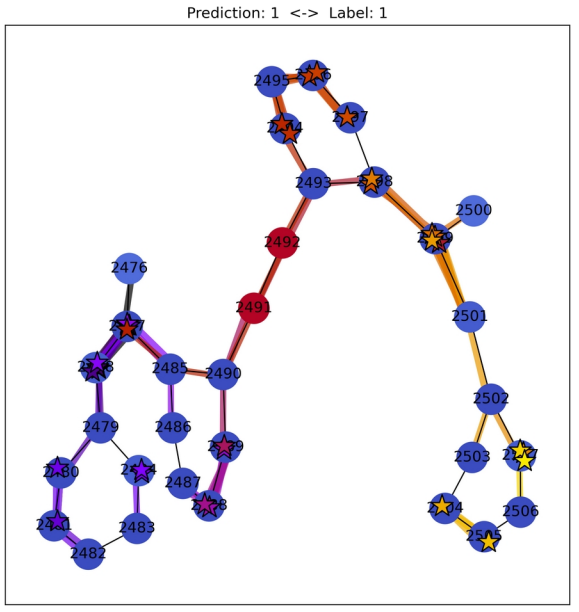


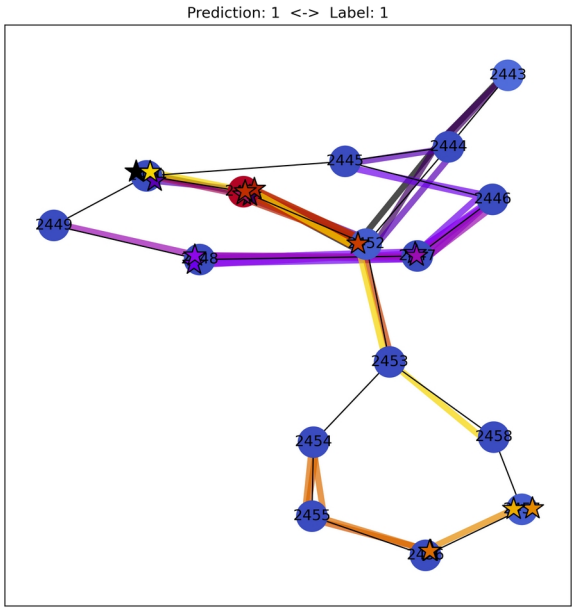


- 1. Part: **Sampling the neighborhood intelligently**
  - We can now sample the neighborhood and optimize the predicted distribution (ST-Gumbel-Softmax)
- 2. Part: **Integrating the Node Features from the path**
  - Using a Sequential Model
- Potential 3. Part: Sampler **Communication**
  - When two samplers are on the same node – exchanging state features would double the receptive field
- Potential 4. Part: Trail Information
  - Comparable to Ant colony optimization, the samplers can leave information at the nodes before they leave

Sampling Trajectories – MolHIV Dataset



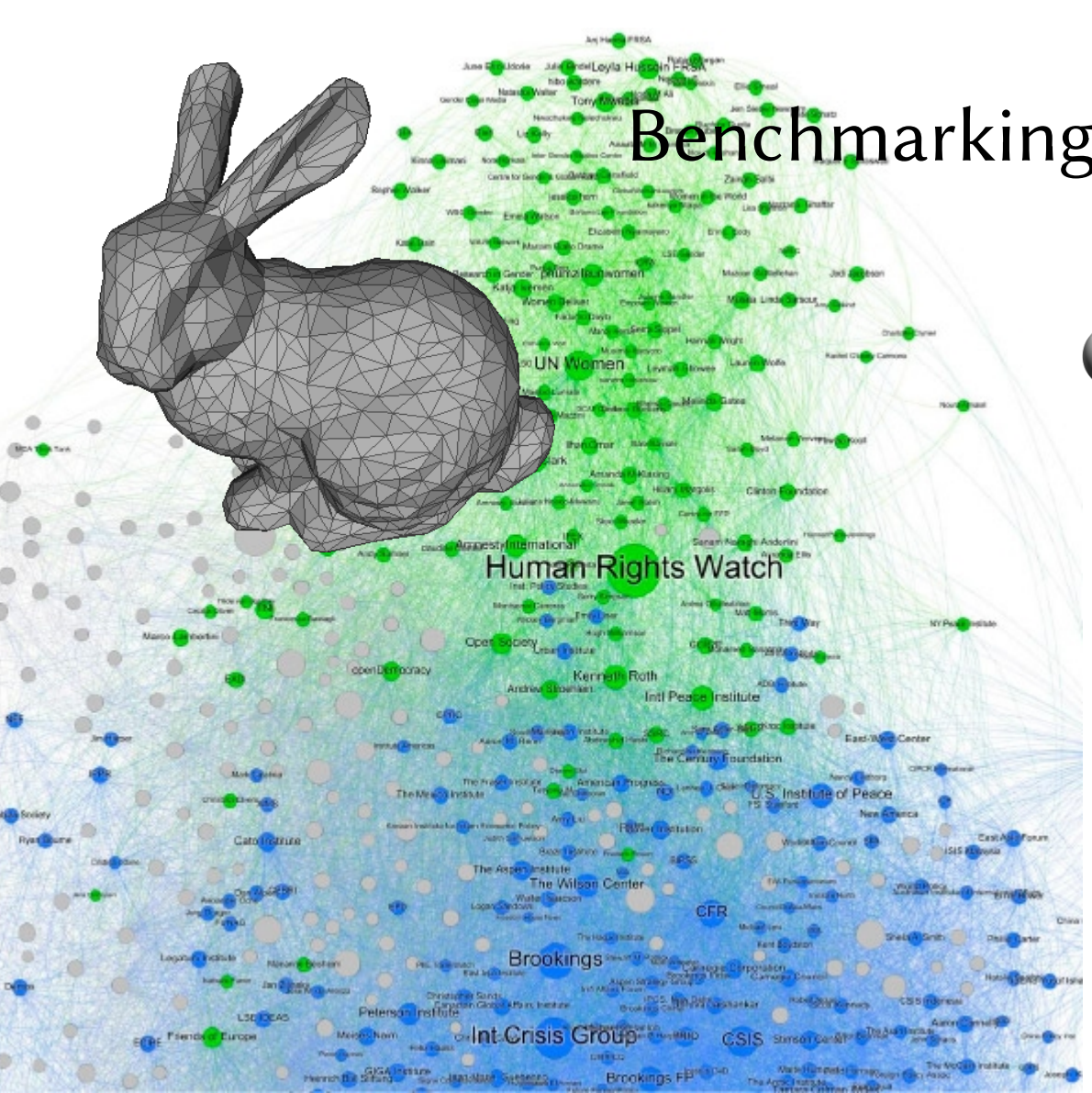


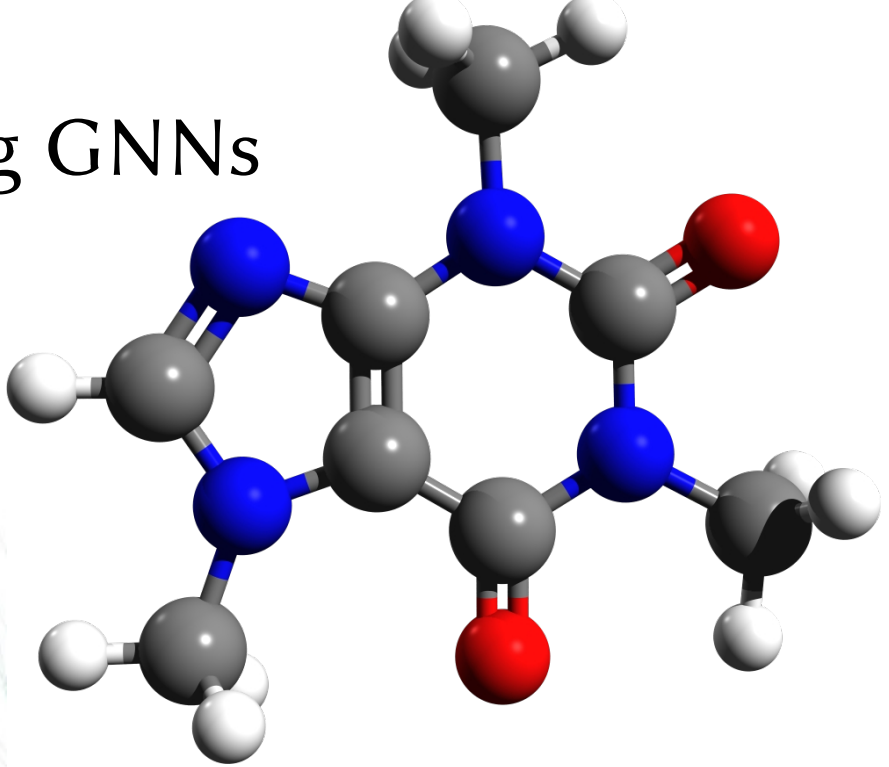


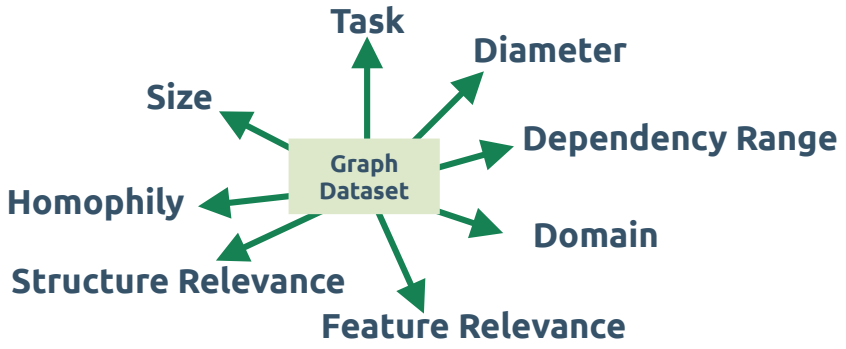
Preliminary Results – Experiments Still Running

Open Question: What's the best way to generate explanations from the sampling trajectories?

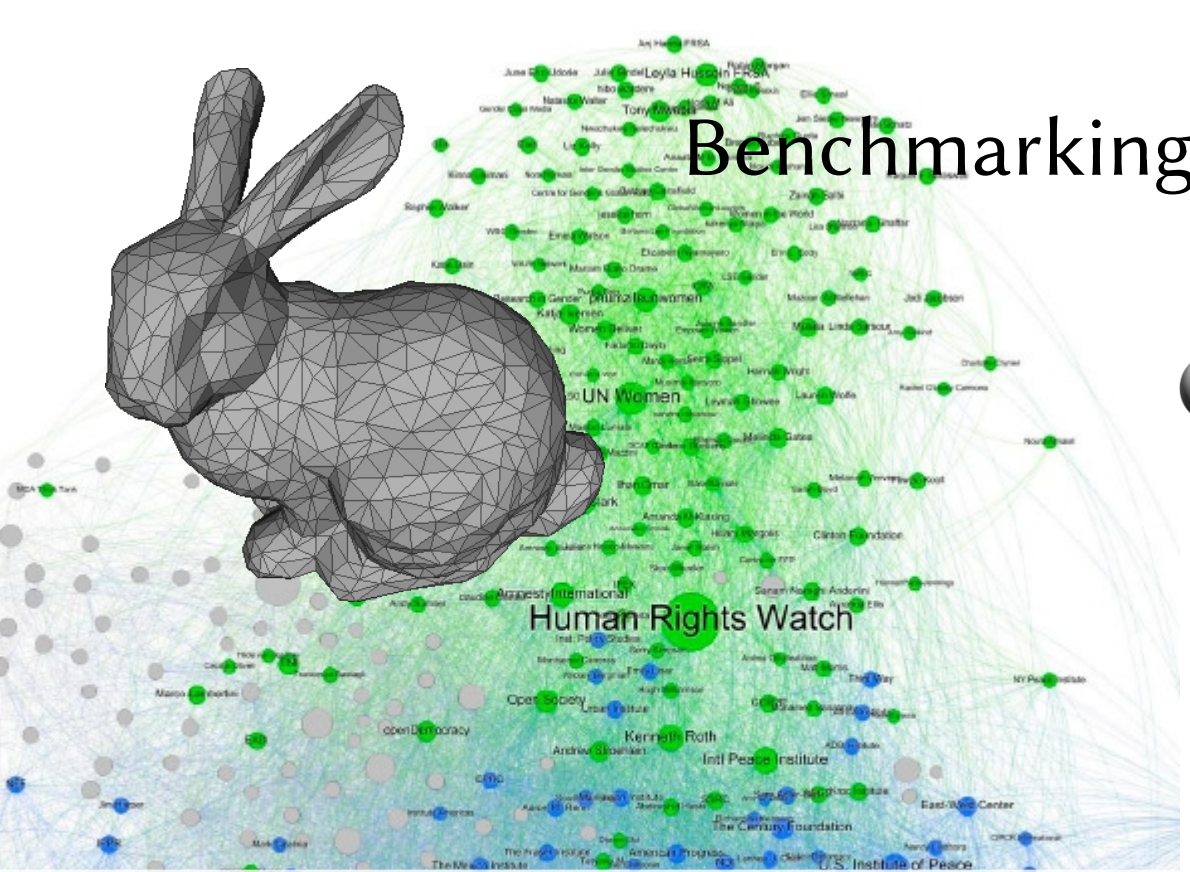
Benchmarking GNNs

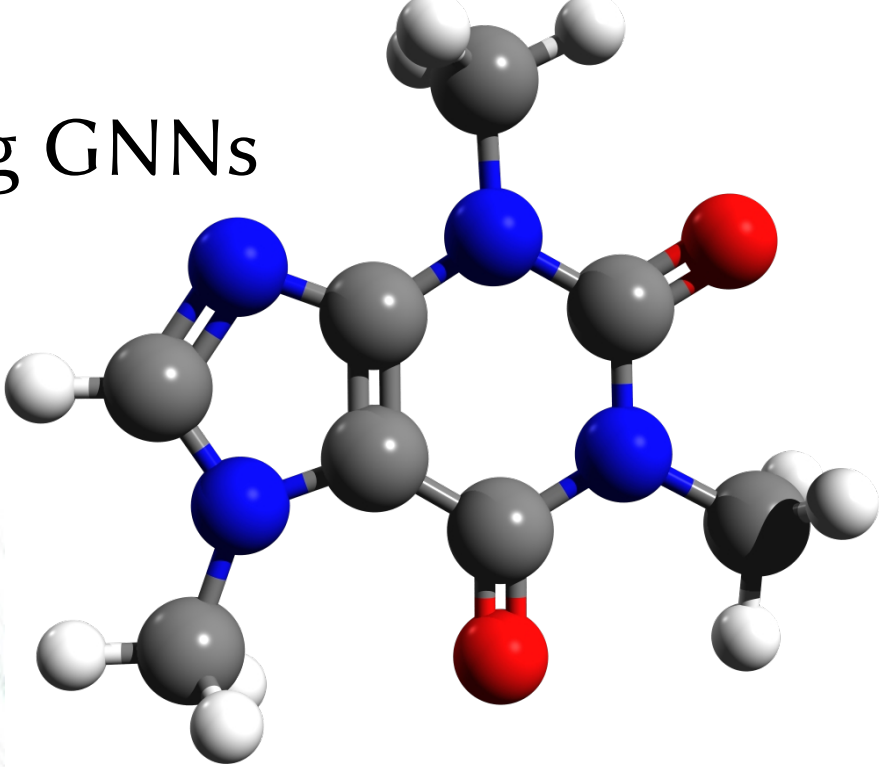


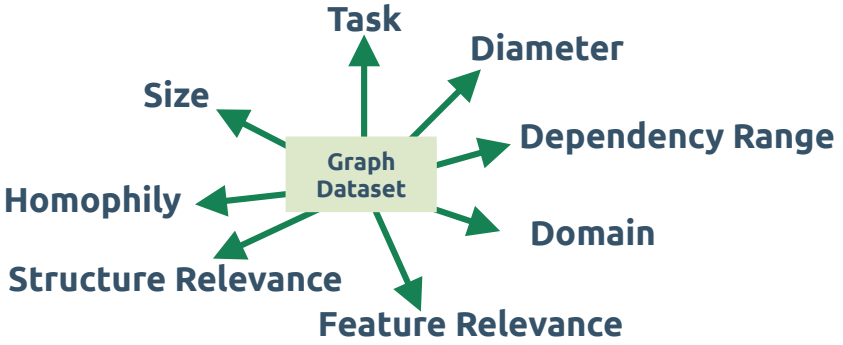




Benchmarking GNNs

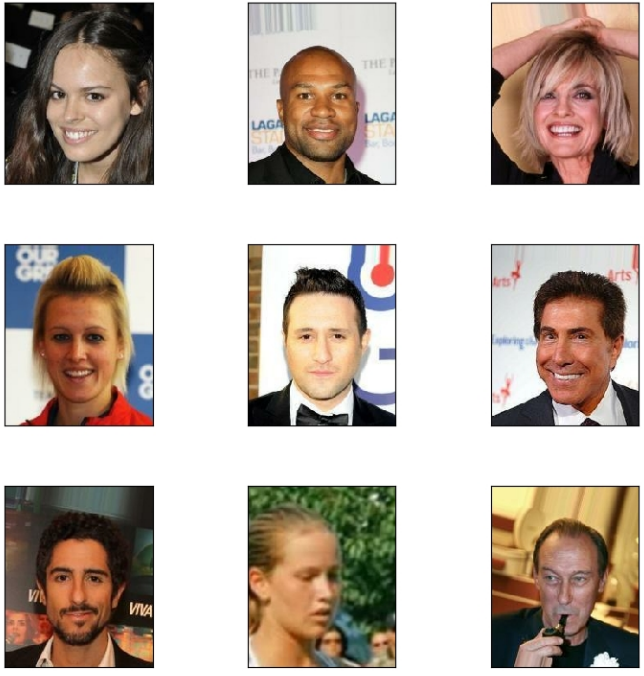






This section focuses on node-level tasks

Image Benchmarks



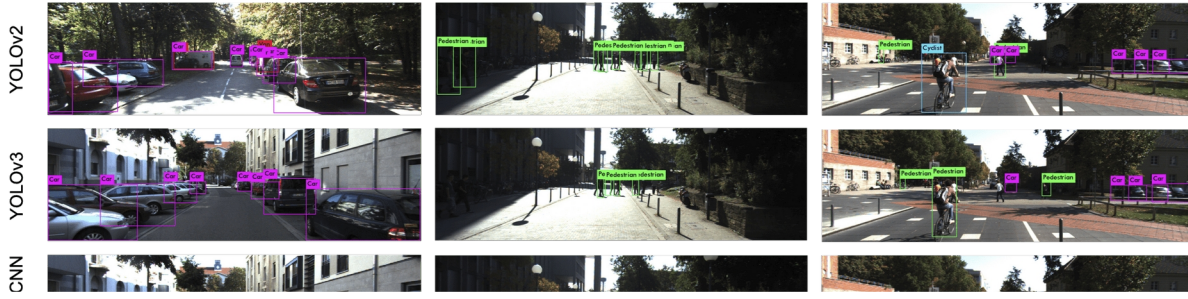


Image Sources

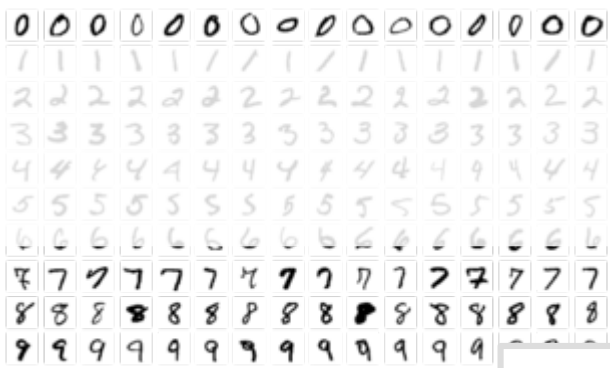
Kitti: https://github.com/topics/kitti-dataset?l=c

ImageNet: https://paperswithcode.com/dataset/imagenet

CelebA: https://www.tensorflow.org/datasets/catalog/celeb_a

MNIST: https://en.wikipedia.org/wiki/MNIST_database

Image Benchmarks



A Model that performs well on ImageNet is likely to also perform well on your own photos, but maybe not suited for dash-cam footage.

→ Different **Domains** / **Sizes** / **Tasks**

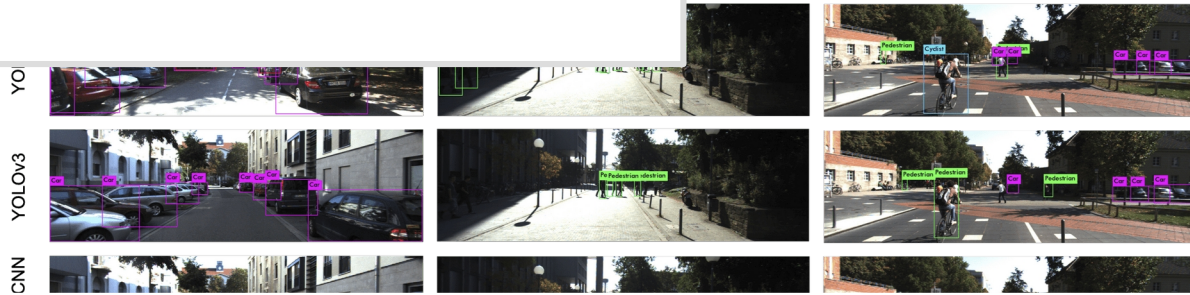


Image Sources

Kitti: https://github.com/topics/kitti-dataset?l=c

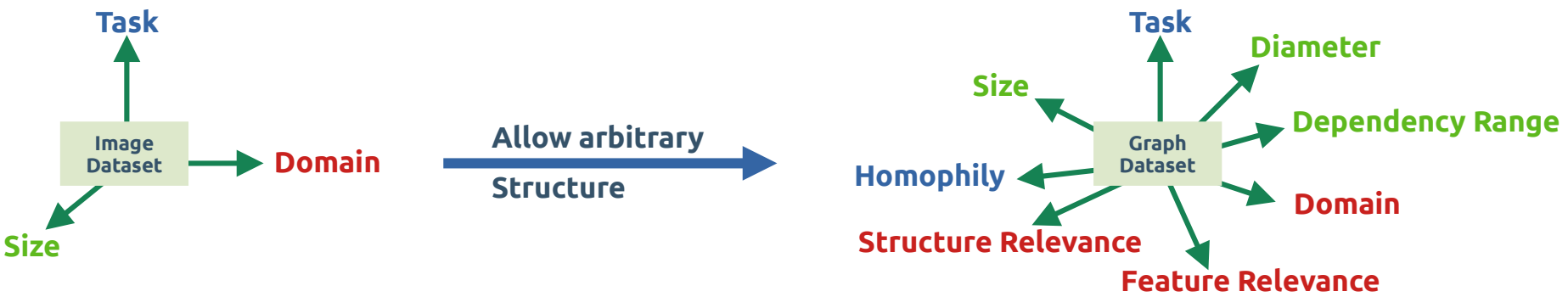
ImageNet: https://paperswithcode.com/dataset/imagenet

CelebA: https://www.tensorflow.org/datasets/catalog/celeb_a

MNIST: https://en.wikipedia.org/wiki/MNIST_database

Benchmarking GNNs

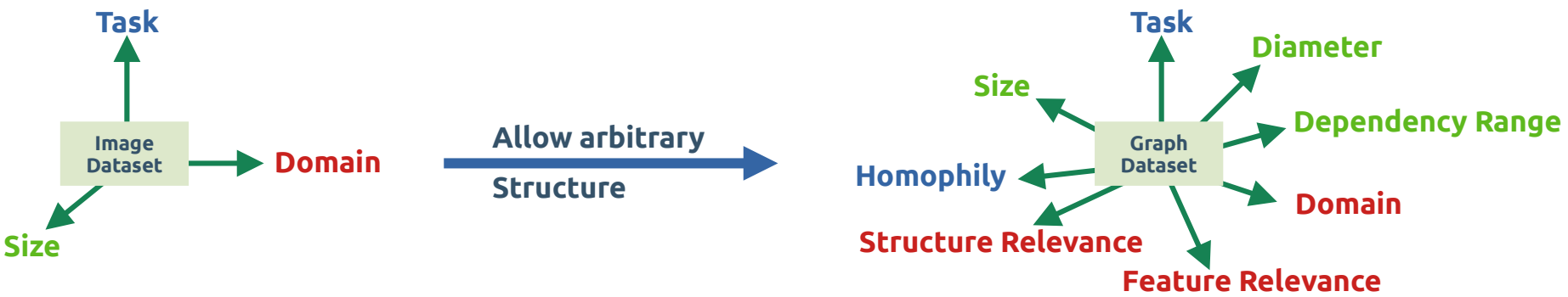
- Variable Structure greatly increases the number of Attributes of a Benchmark
- It is less intuitive what model suits which need



Benchmarking GNNs

- Variable Structure greatly increases the number of Attributes of a Benchmark
- It is less intuitive what model suits which need

  → **Still an open question and not well understood**



Benchmarking GNNs

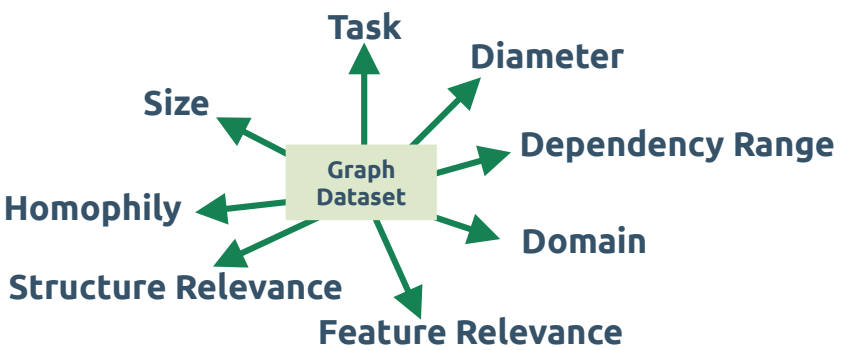
- A typical table header at the end of a paper:

Model	Cora	Pubmed	Citeseer
Some baselines	...	...	...
Some Related Work	...	...	...
Presented Work	...	...	...

The MNISTs of Graphs: Citation Networks

Table 1: Average node classification test accuracy results over X seeds.

- What do these benchmarks tell us about a model?



Pitfalls of Relying on Cora and Co.

- In this benchmark, GCN performance seems to correlate with *homophily*
  - The tendency that edges connect similar nodes
- This aligns well with the smoothing property which might explain the difference in performance

Table 1: Results on node classification datasets sorted by their homophily level. Top three models are coloured by **First**, **Second**, **Third**. Our models are marked **NSD**.

	Texas	**Wisconsin**	**Film**	**Squirrel**	**Chameleon**	**Cornell**	**Citeseer**	**Pubmed**	**Cora**
Hom level	**0.11**	**0.21**	**0.22**	**0.22**	**0.23**	**0.30**	**0.74**	**0.80**	**0.81**
#Nodes	183	251	7,600	5,201	2,277	183	3,327	18,717	2,708
#Edges	295	466	26,752	198,493	31,421	280	4,676	44,327	5,278
#Classes	5	5	5	5	5	5	7	3	6
GraphSAGE	82.43±6.14	81.18±5.56	34.23±0.99	41.61±0.74	58.73±1.68	75.95±5.01	76.04±1.30	88.45±0.50	86.90±1.04
GCN	55.14±5.16	51.76±3.06	27.32±1.10	53.43±2.01	64.82±2.24	60.54±5.30	76.50±1.36	88.42±0.50	86.98±1.27
GAT	52.16±6.63	49.41±4.09	27.44±0.89	40.72±1.55	60.26±2.50	61.89±5.05	76.55±1.23	87.30±1.10	86.33±0.48
MLP	80.81±4.75	85.29±3.31	36.53±0.70	28.77±1.56	46.21±2.99	81.89±6.40	74.02±1.90	87.16±0.37	75.69±2.00

Two Spatial Graph Models

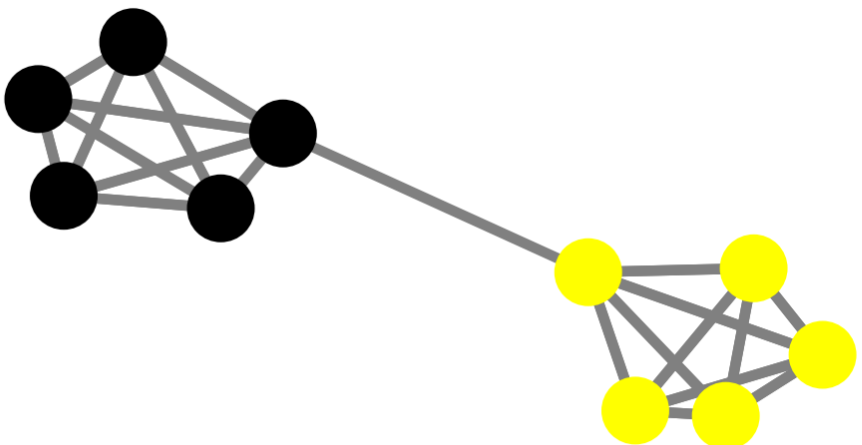
Tabel derived from Bodnar et al. (2022) Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs

Homophily - Heterophily

- Tendency that edges connect similar nodes
- Not formalized, different ways to compute this
- A simple formulation:

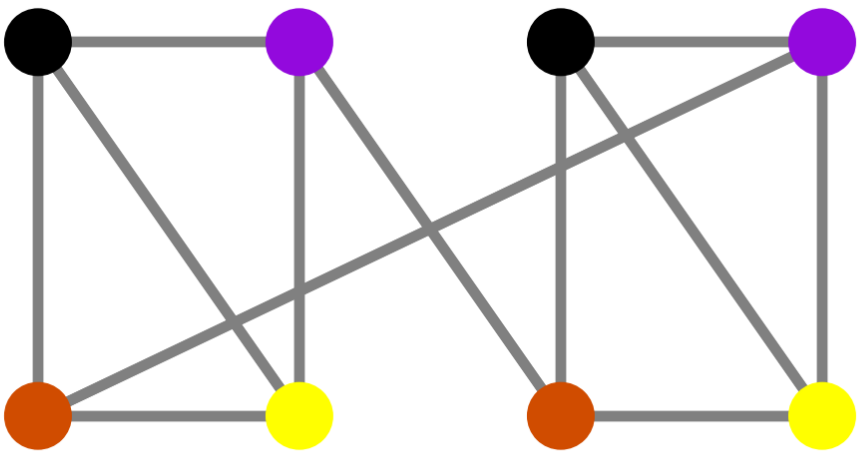
$$H(V, E) = \frac{|(u, v) \in E : y_u = y_v|}{|E|}$$

Homophile Graph



H = 0.98

Heterophile Graph



H = 0.0

Pitfalls of Benchmarking GNNs [1]

	Relative accuracy	Avg. rank
GCN	99.4	2.3
MoNet	99.0	2.7
GS-mean	98.3	2.7
GAT	95.9	3.6
GS-meanpool	93.0	5.2
GS-maxpool	91.1	6.4
LabelProp NL	89.3	7.4
LabelProp	86.6	7.7
LogReg	80.6	8.8
MLP	77.8	8.8

(a) Relative accuracy and average rank.

Planetoid split	**CORA**	**CiteSeer**	**PubMed**
GCN	81.9 ± 0.8	69.5 ± 0.9	**79.0** ± 0.5
GAT	**82.8** ± 0.5	**71.0** ± 0.6	77.0 ± 1.3
MoNet	82.2 ± 0.7	70.0 ± 0.6	77.7 ± 0.6
GS-maxpool	77.4 ± 1.0	67.0 ± 1.0	76.6 ± 0.8
Another split	**CORA**	**CiteSeer**	**PubMed**
GCN	**79.0** ± 0.7	**68.6** ± 1.1	69.5 ± 1.0
GAT	77.9 ± 0.7	67.7 ± 1.2	69.5 ± 0.6
MoNet	77.9 ± 0.7	66.8 ± 1.3	**70.7** ± 0.5
GS-maxpool	74.5 ± 0.6	63.1 ± 1.2	70.3 ± 0.8

(b) Different split leads to a completely different ranking of models.

Table 2: (a) Relative accuracy scores and ranks averaged over all datasets. See text for the definition. (b) Model accuracy on the Planetoid split from Yang et al. [2016] and another split on the same datasets. Different splits lead to a completely different ranking of models.

Different Data-Splits can lead to significantly different results

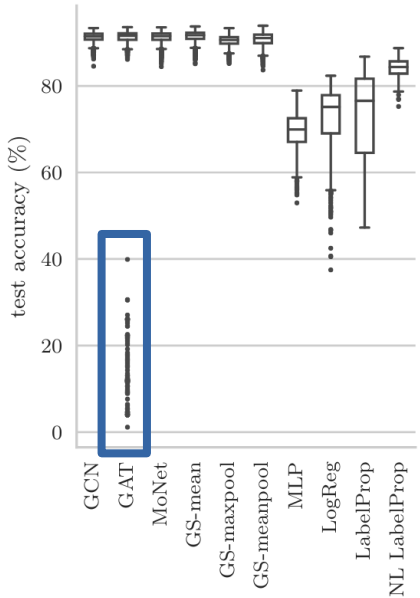
→ Simpler Architectures even outperform more sophisticated ones

→ Usually the data splits are the same across papers

[1] Oleksandr et al. (2019) Pitfalls of Graph Neural Network Evaluation

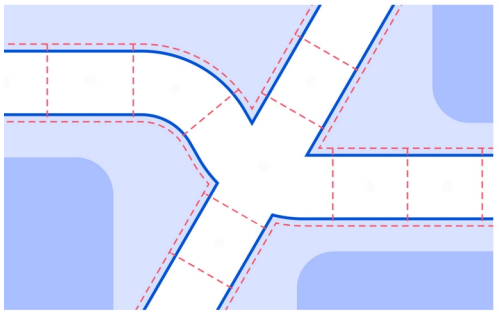
Pitfalls of Benchmarking GNNs [1]

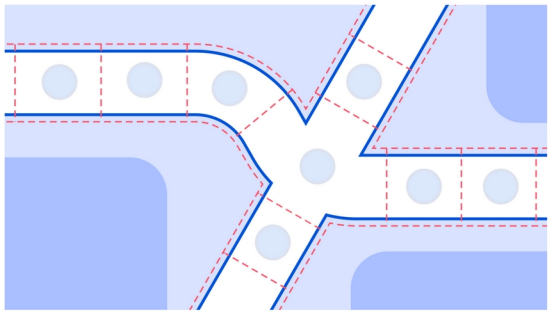
- Another finding:
  Different weight initialization methods can produce vastly different results as well
- A complete hyperparameter search with cross validation for each configuration?

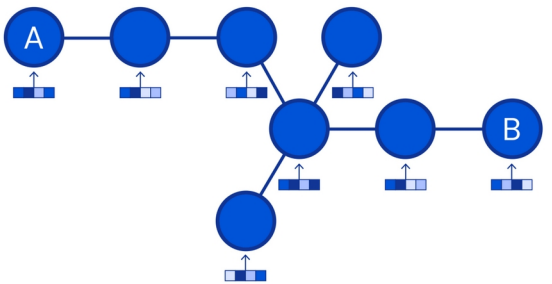


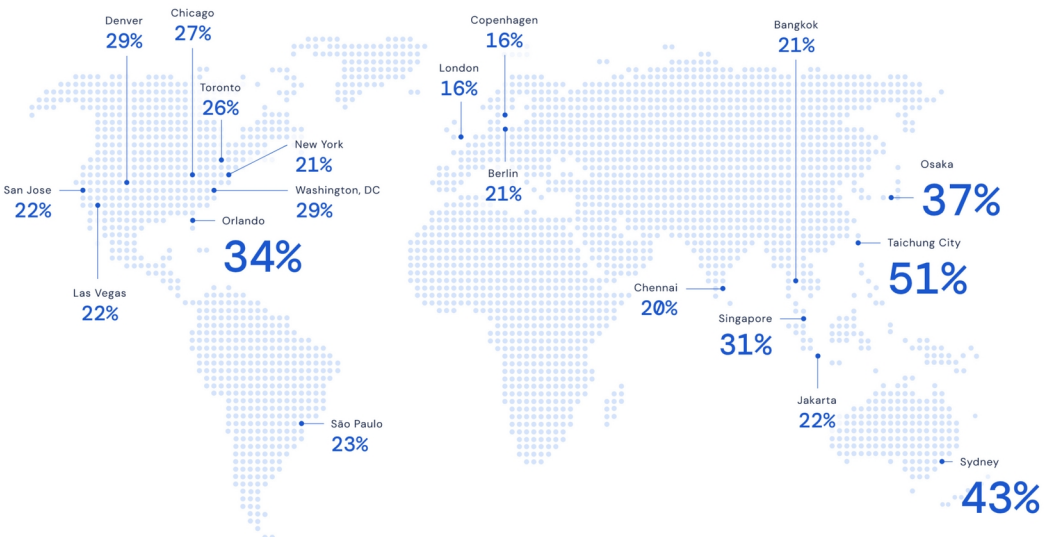
[1] Oleksandr et al. (2019) Pitfalls of Graph Neural Network Evaluation

Google's ETA Prediction









Derrow-Pinion et al. (2021) ETA Prediction with Graph Neural Networks in Google Maps

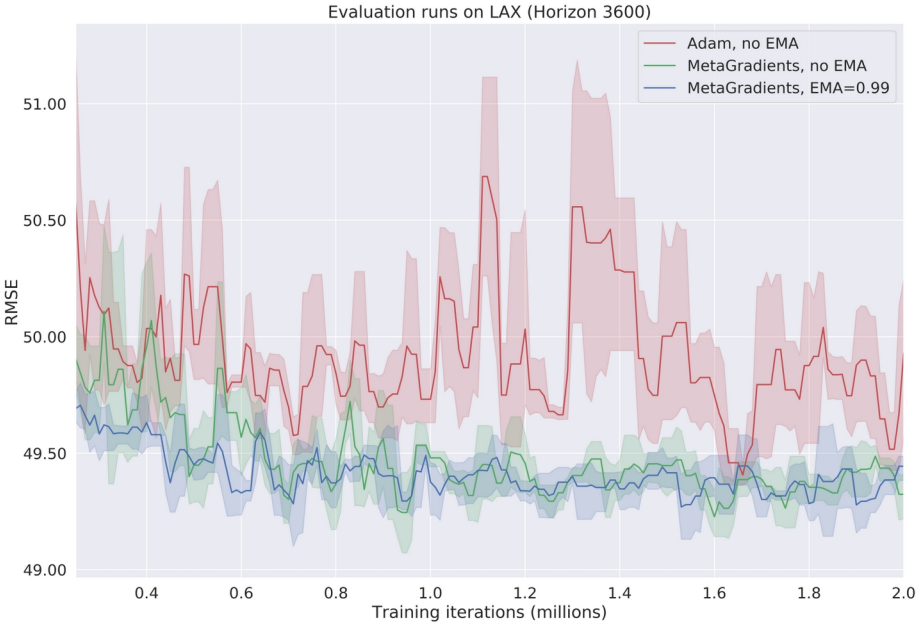


Figure 3: Validation RMSE during training, with and without MetaGradients and EMA decay, aggregated across five seeds. Both methods contribute to variance reduction. Results shown for two metro/horizon setups – same trends hold elsewhere.

> Applying stabilising techniques such as **MetaGradients and EMA [(Exponential Moving Average)]** was a necessary addition to make the GNNs production-ready.

Derrow-Pinion et al. (2021) ETA Prediction with Graph Neural Networks in Google Maps

I asked ChatGPT

LH

What aspects of benchmarking graph neural networks are important?

Benchmarking graph neural networks is an important part of evaluating their performance and comparing different models. Some of the key aspects to consider when benchmarking graph neural networks include the quality of the graph data used for training and evaluation, the specific tasks and metrics used to evaluate the performance of the model, and the computational resources required to train and evaluate the model. Additionally, it is important to consider the scalability of the model and its ability to generalize to new, unseen data.

LH

What are the pitfalls on GNN evaluation?

There are several potential pitfalls to be aware of when evaluating the performance of a graph neural network (GNN). Some of the most common pitfalls include the following:

1. Using the wrong evaluation metric: It is important to use evaluation metrics that are appropriate for the specific task and type of data being considered. For example, accuracy may be a good metric for evaluating a GNN that is used for node classification, but it may not be appropriate for evaluating a GNN that is used for link prediction.
2. Failing to account for the underlying structure of the graph: GNNs are designed to take advantage of the structure of the graph data they operate on, so it is important to carefully consider how this structure affects the performance of the GNN. For example, a GNN may perform well on a particular type of graph, but poorly on a different type of graph with a different structure.
3. Failing to account for the diversity of the data: The performance of a GNN may vary depending on the diversity of the data it is applied to. It is important to evaluate a GNN on a diverse set of data to ensure that it can generalize to different types of graphs.
4. Failing to account for the size of the graph: The performance of a GNN may vary depending on the size of the graph it is applied to. It is important to evaluate a GNN on a range of graph sizes to ensure that it can handle graphs of different sizes.

Overall, it is important to carefully consider these and other potential pitfalls when evaluating the performance of a GNN, in order to accurately assess its capabilities and limitations.

LH

What are the pitfalls on GNN evaluation?

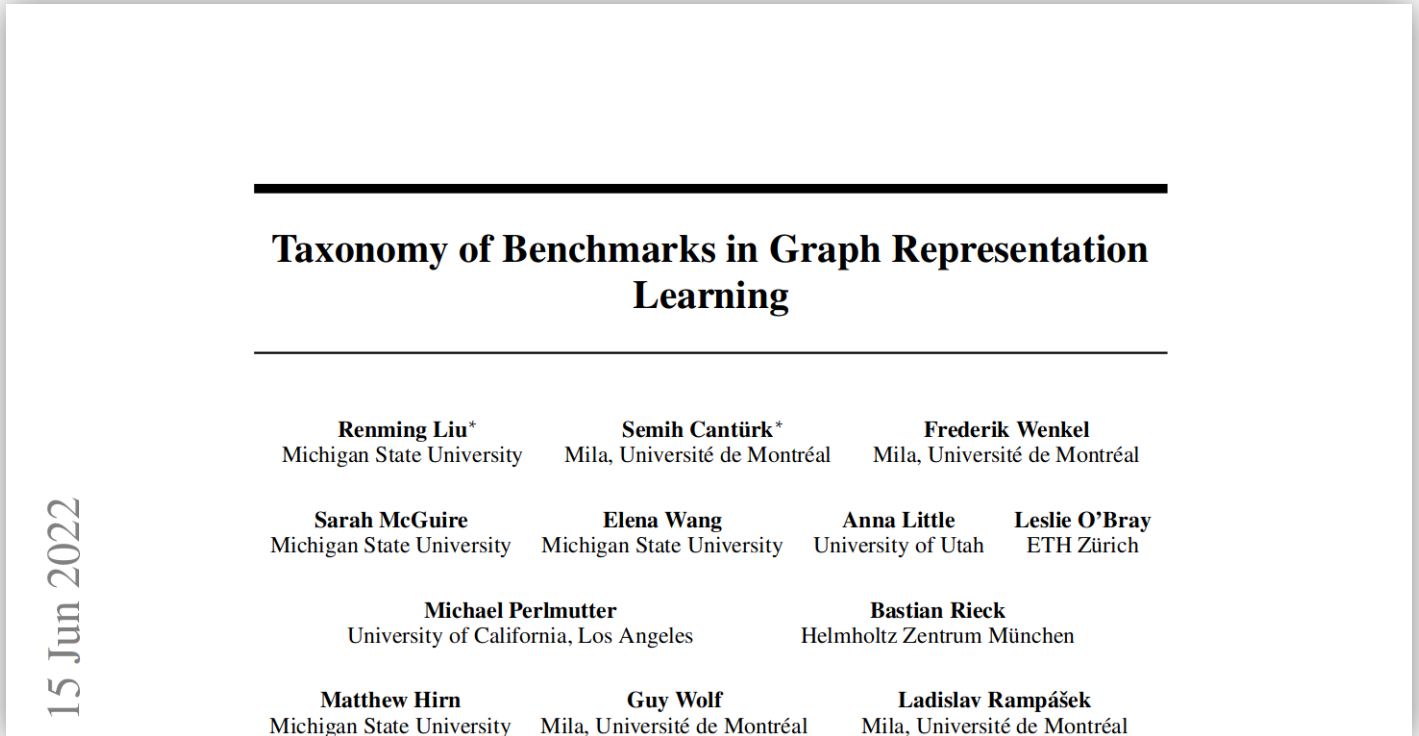
There are several potential pitfalls to be aware of whe
graph neural network (GNN). Some of the most comm

1. Using the wrong evaluation metric: It is important t
appropriate for the specific task and type of data being con ample,
accuracy may be a good metric for evaluating a GNN that i ode
classification, but it may not be appropriate for evaluatin that is used for link
prediction.
2. Failing to account for the underlying structure of the aph: GNNs are designed to take advantage of the structure of the graph data they operate on, so it is important to carefully consider how this structure affects the performance of the GNN. For example, a GNN may perform well on a particular type of graph, but poorly on a different type of graph with a different structure.
3. Failing to account for the diversity of the data: The performance of a GNN may vary depending on the diversity of the data it is applied to. It is important to evaluate a GNN on a diverse set of data to ensure that it can generalize to different types of graphs.
4. Failing to account for the size of the graph: The performance of a GNN may vary depending on the size of the graph it is applied to. It is important to evaluate a GNN on a range of graph sizes to ensure that it can handle graphs of different sizes.

Overall, it is important to carefully consider these and other potential pitfalls when evaluating the performance of a GNN, in order to accurately assess its capabilities and limitations.

There is not really a good way to quantify this. Also, we have features AND structure.

Jointly Benchmarking Datasets and Models



Taxonomy of Benchmarks in Graph Representation Learning

Renming Liu[*]
Michigan State University

Semih Cantürk[*]
Mila, Université de Montréal

Frederik Wenkel
Mila, Université de Montréal

Sarah McGuire
Michigan State University

Elena Wang
Michigan State University

Anna Little
University of Utah

Leslie O'Bray
ETH Zürich

Michael Perlmutter
University of California, Los Angeles

Bastian Rieck
Helmholtz Zentrum München

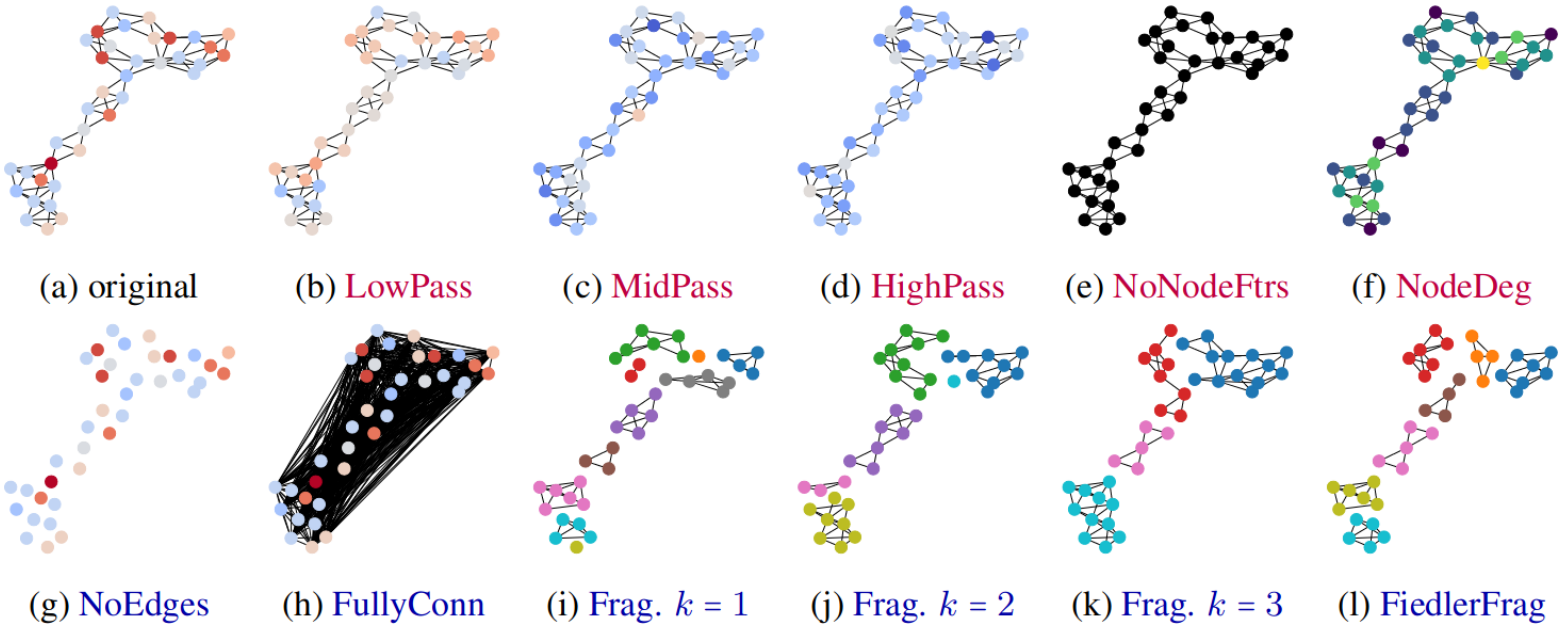
Matthew Hirn
Michigan State University

Guy Wolf
Mila, Université de Montréal

Ladislav Rampášek
Mila, Université de Montréal

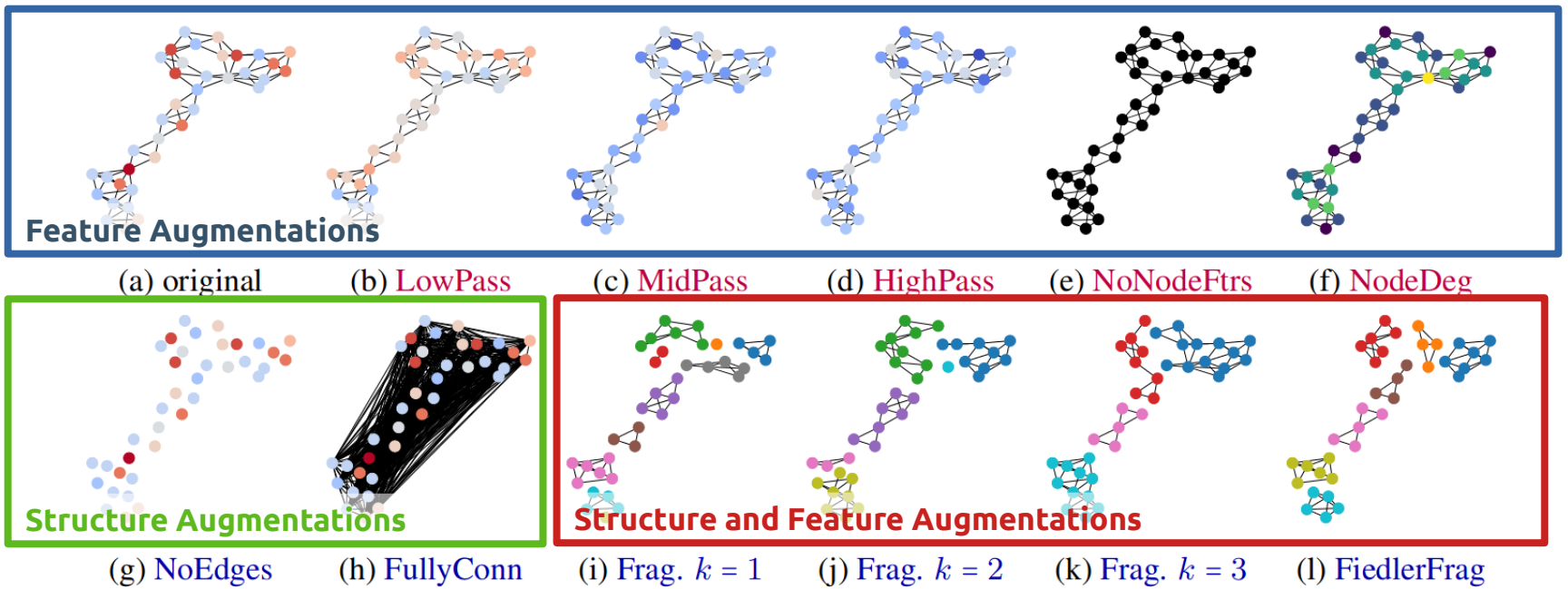
Graph Perturbations

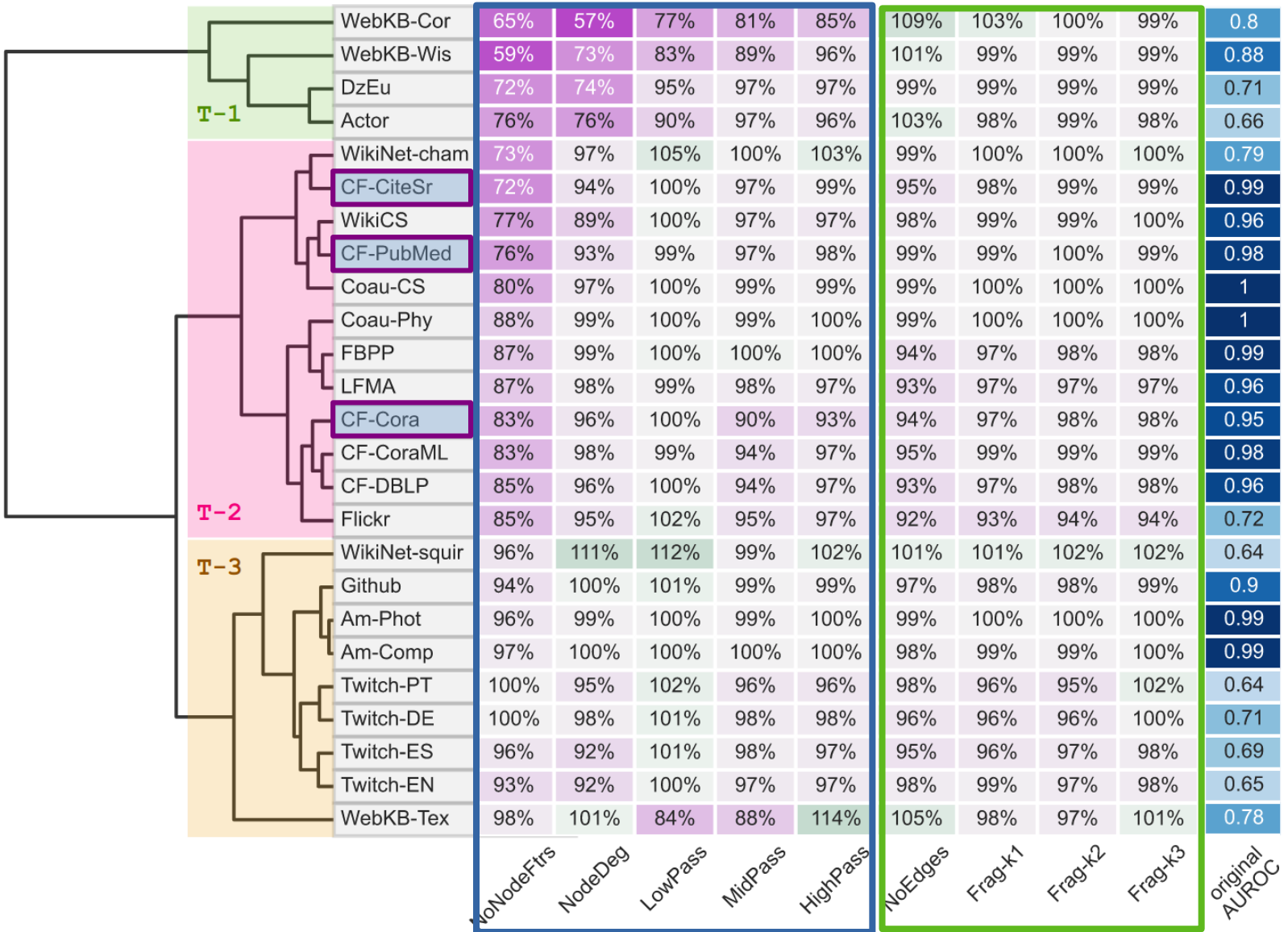
- Perturb datasets by several means and look at the performance



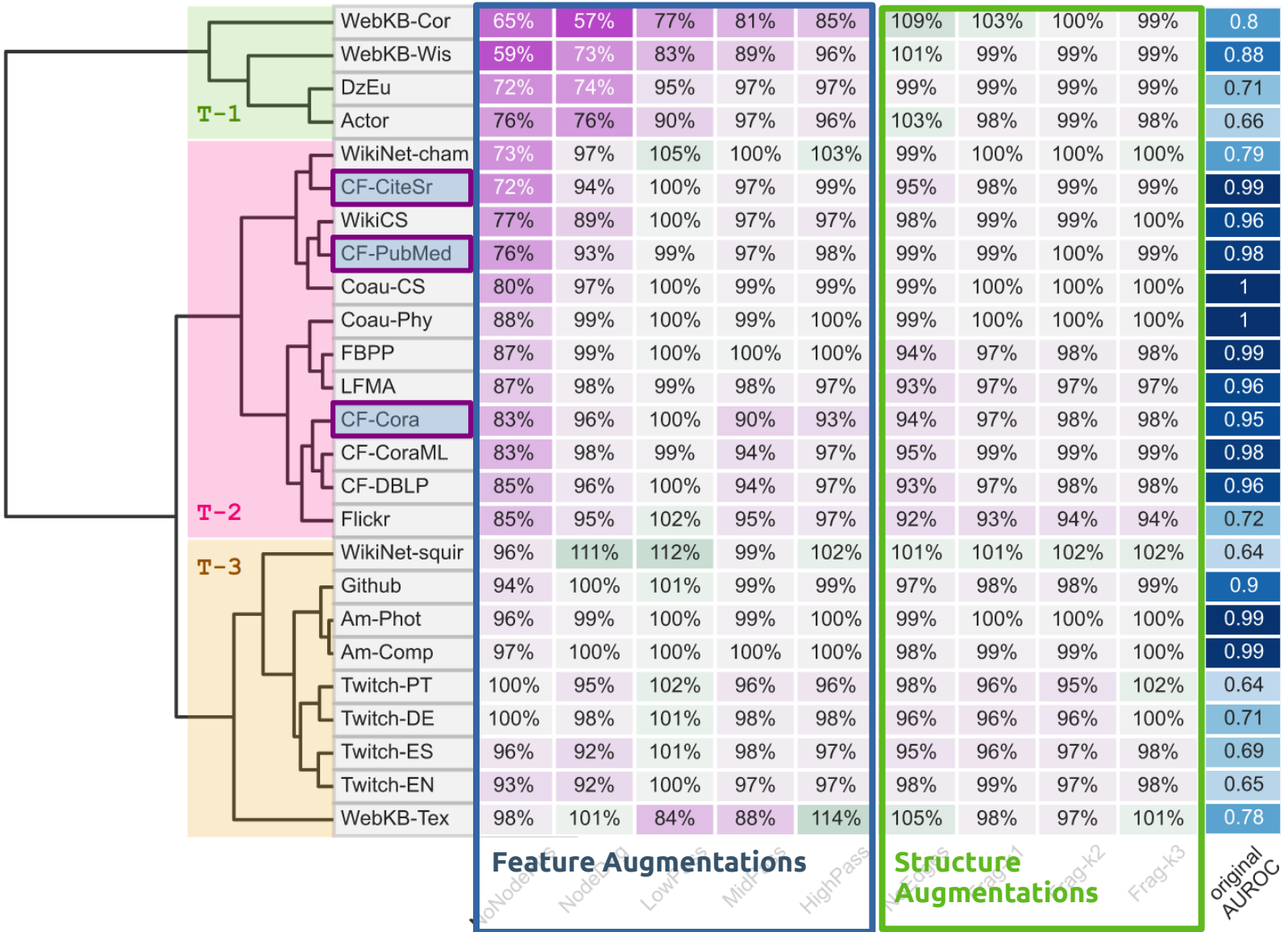
Graph Perturbations

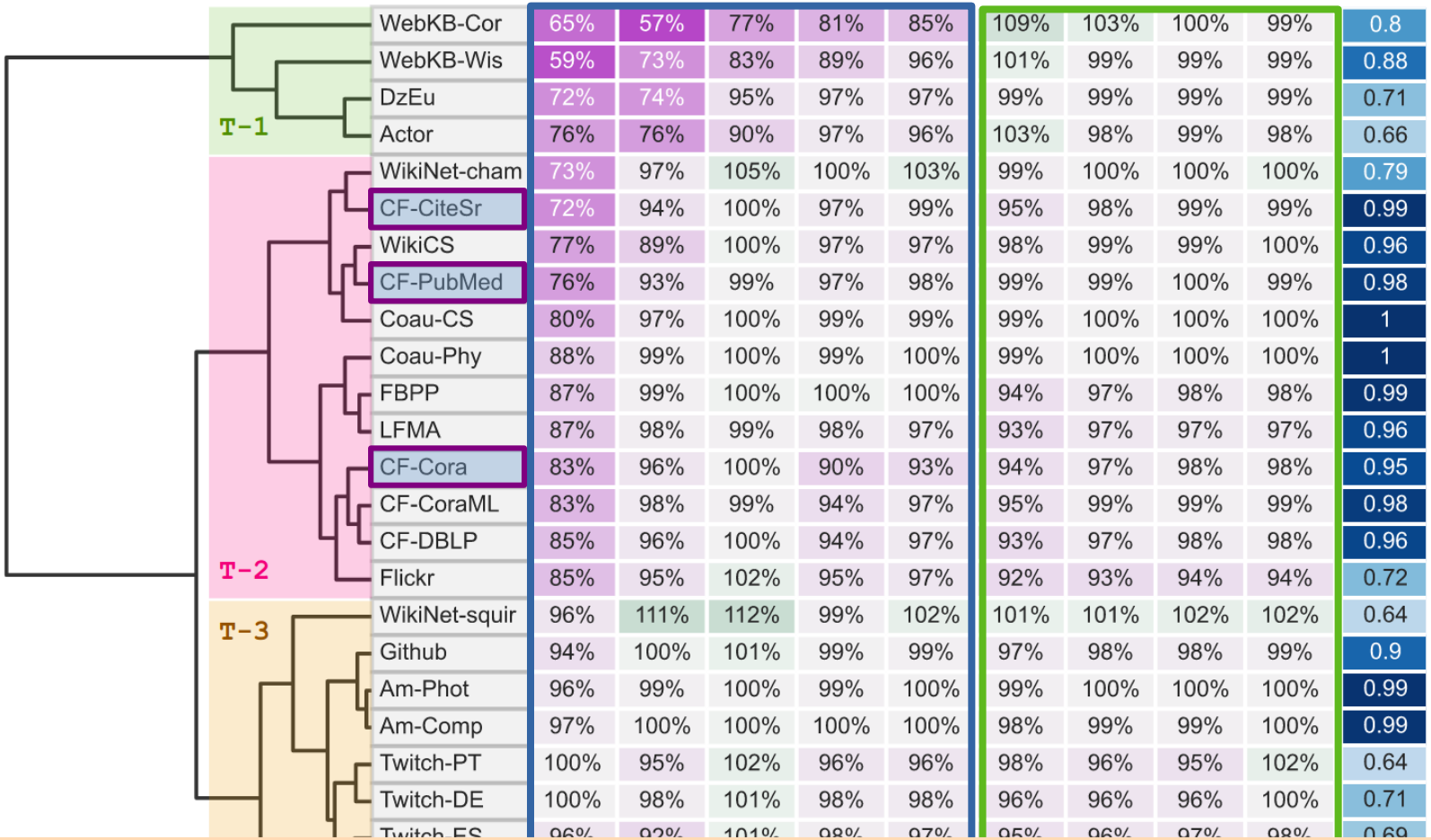
- Perturb datasets by several means and look at the performance



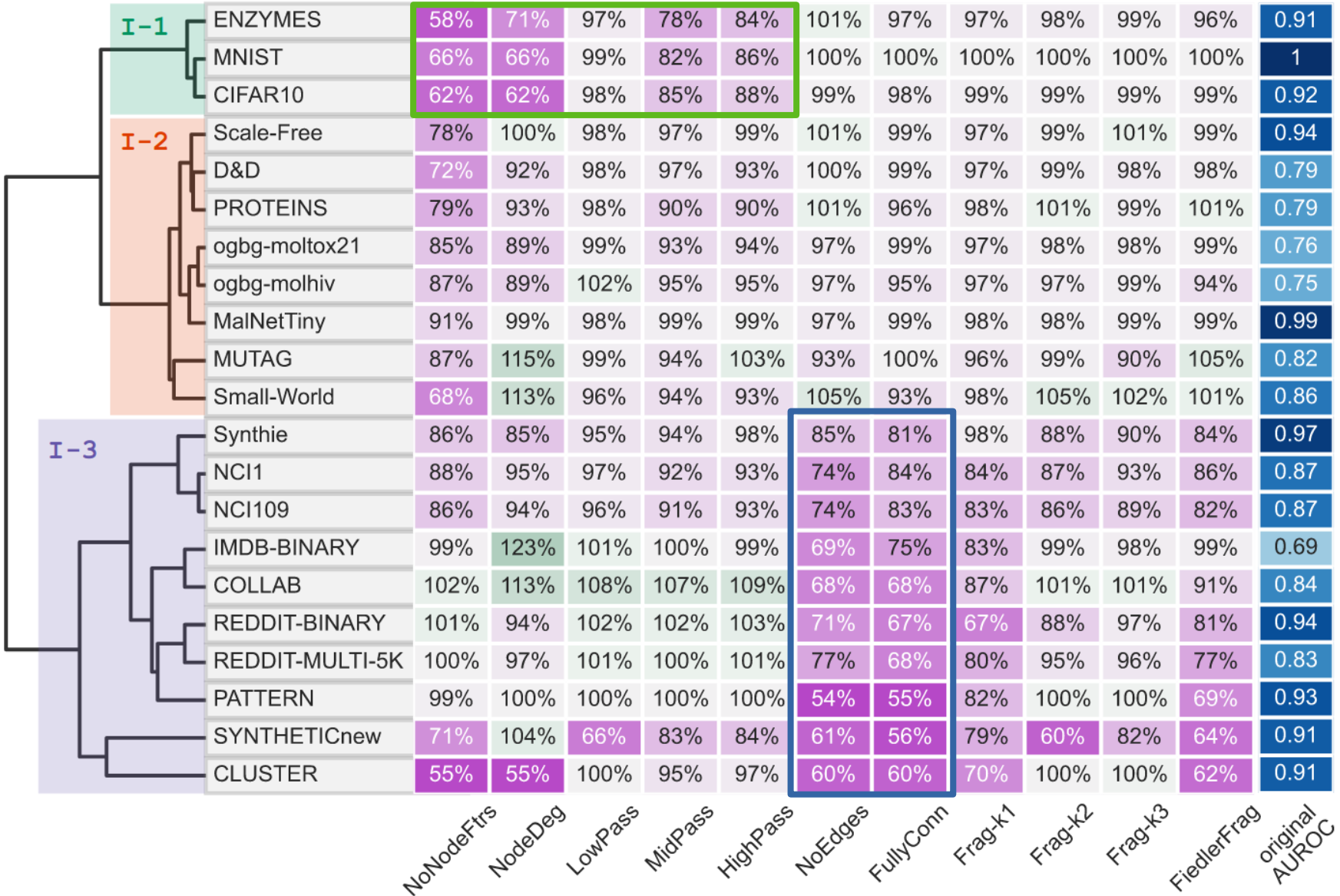


	NoNodeFtrs	NodeDeg	LowPass	MidPass	HighPass	NoEdges	Frag-k1	Frag-k2	Frag-k3	original AUROC
WebKB-Cor	65%	57%	77%	81%	85%	109%	103%	100%	99%	0.8
WebKB-Wis	59%	73%	83%	89%	96%	101%	99%	99%	99%	0.88
DzEu	72%	74%	95%	97%	97%	99%	99%	99%	99%	0.71
Actor	76%	76%	90%	97%	96%	103%	98%	99%	98%	0.66
WikiNet-cham	73%	97%	105%	100%	103%	99%	100%	100%	100%	0.79
CF-CiteSr	72%	94%	100%	97%	99%	95%	98%	99%	99%	0.99
WikiCS	77%	89%	100%	97%	97%	98%	99%	99%	100%	0.96
CF-PubMed	76%	93%	99%	97%	98%	99%	99%	100%	99%	0.98
Coau-CS	80%	97%	100%	99%	99%	99%	100%	100%	100%	1
Coau-Phy	88%	99%	100%	99%	100%	99%	100%	100%	100%	1
FBPP	87%	99%	100%	100%	100%	94%	97%	98%	98%	0.99
LFMA	87%	98%	99%	98%	97%	93%	97%	97%	97%	0.96
CF-Cora	83%	96%	100%	90%	93%	94%	97%	98%	98%	0.95
CF-CoraML	83%	98%	99%	94%	97%	95%	99%	99%	99%	0.98
CF-DBLP	85%	96%	100%	94%	97%	93%	97%	98%	98%	0.96
Flickr	85%	95%	102%	95%	97%	92%	93%	94%	94%	0.72
WikiNet-squir	96%	111%	112%	99%	102%	101%	101%	102%	102%	0.64
Github	94%	100%	101%	99%	99%	97%	98%	98%	99%	0.9
Am-Phot	96%	99%	100%	99%	100%	99%	100%	100%	100%	0.99
Am-Comp	97%	100%	100%	100%	100%	98%	99%	99%	100%	0.99
Twitch-PT	100%	95%	102%	96%	96%	98%	96%	95%	102%	0.64
Twitch-DE	100%	98%	101%	98%	98%	96%	96%	96%	100%	0.71
Twitch-ES	96%	92%	101%	98%	97%	95%	96%	97%	98%	0.69
Twitch-EN	93%	92%	100%	97%	97%	98%	99%	97%	98%	0.65
WebKB-Tex	98%	101%	84%	88%	114%	105%	98%	97%	101%	0.78


T-1
T-2
T-3
WebKB-Cor 65% 57% 77% 81% 85% 109% 103% 100% 99% 0.8
WebKB-Wis 59% 73% 83% 89% 96% 101% 99% 99% 99% 0.88
DzEu 72% 74% 95% 97% 97% 99% 99% 99% 99% 0.71
Actor 76% 76% 90% 97% 96% 103% 98% 99% 98% 0.66
WikiNet-cham 73% 97% 105% 100% 103% 99% 100% 100% 100% 0.79
CF-CiteSr 72% 94% 100% 97% 99% 95% 98% 99% 99% 0.99
WikiCS 77% 89% 100% 97% 97% 98% 99% 99% 100% 0.96
CF-PubMed 76% 93% 99% 97% 98% 99% 99% 100% 99% 0.98
Coau-CS 80% 97% 100% 99% 99% 99% 100% 100% 100% 1
Coau-Phy 88% 99% 100% 99% 100% 99% 100% 100% 100% 1
FBPP 87% 99% 100% 100% 100% 94% 97% 98% 98% 0.99
LFMA 87% 98% 99% 98% 97% 93% 97% 97% 97% 0.96
CF-Cora 83% 96% 100% 90% 93% 94% 97% 98% 98% 0.95
CF-CoraML 83% 98% 99% 94% 97% 95% 99% 99% 99% 0.98
CF-DBLP 85% 96% 100% 94% 97% 93% 97% 98% 98% 0.96
Flickr 85% 95% 102% 95% 97% 92% 93% 94% 94% 0.72
WikiNet-squir 96% 111% 112% 99% 102% 101% 101% 102% 102% 0.64
Github 94% 100% 101% 99% 99% 97% 98% 98% 99% 0.9
Am-Phot 96% 99% 100% 99% 100% 99% 100% 100% 100% 0.99
Am-Comp 97% 100% 100% 100% 100% 98% 99% 99% 100% 0.99
Twitch-PT 100% 95% 102% 96% 96% 98% 96% 95% 102% 0.64
Twitch-DE 100% 98% 101% 98% 98% 96% 96% 96% 100% 0.71
Twitch-ES 96% 92% 101% 98% 97% 95% 96% 97% 98% 0.69
Twitch-EN 93% 92% 100% 97% 97% 98% 99% 97% 98% 0.65
WebKB-Tex 98% 101% 84% 88% 114% 105% 98% 97% 101% 0.78
Feature Augmentations
Structure Augmentations
HighPass
Frag-k3
original AUROC



- Some benchmarks heavily rely on informative node features
- Many Benchmarks don't actually focus heavily on structure.
- Authors Propose to benchmark on Datasets from each of these clusters T-1, T-2, T-3

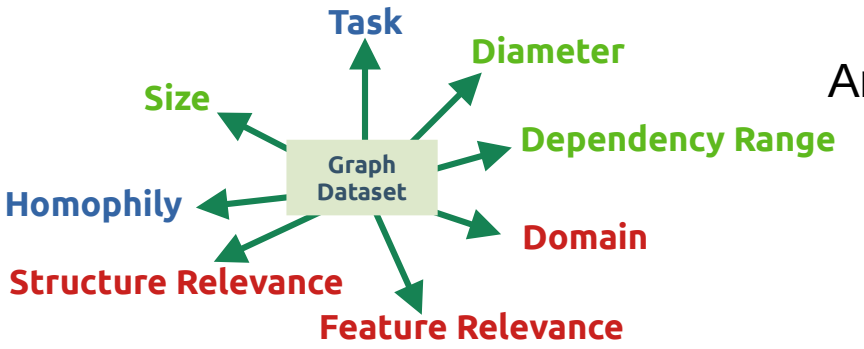

I-1
I-2
I-3
ENZYMES 58% 71% 97% 78% 84% 101% 97% 97% 98% 99% 96% 0.91
MNIST 66% 66% 99% 82% 86% 100% 100% 100% 100% 100% 100% 1
CIFAR10 62% 62% 98% 85% 88% 99% 98% 99% 99% 99% 99% 0.92
Scale-Free 78% 100% 98% 97% 99% 101% 99% 97% 99% 101% 99% 0.94
D&D 72% 92% 98% 97% 93% 100% 99% 97% 99% 98% 98% 0.79
PROTEINS 79% 93% 98% 90% 90% 101% 96% 98% 101% 99% 101% 0.79
ogbg-moltox21 85% 89% 99% 93% 94% 97% 99% 97% 98% 98% 99% 0.76
ogbg-molhiv 87% 89% 102% 95% 95% 97% 95% 97% 97% 99% 94% 0.75
MalNetTiny 91% 99% 98% 99% 99% 97% 99% 98% 98% 99% 99% 0.99
MUTAG 87% 115% 99% 94% 103% 93% 100% 96% 99% 90% 105% 0.82
Small-World 68% 113% 96% 94% 93% 105% 93% 98% 105% 102% 101% 0.86
Synthie 86% 85% 95% 94% 98% 85% 81% 98% 88% 90% 84% 0.97
NCI1 88% 95% 97% 92% 93% 74% 84% 84% 87% 93% 86% 0.87
NCI109 86% 94% 96% 91% 93% 74% 83% 83% 86% 89% 82% 0.87
IMDB-BINARY 99% 123% 101% 100% 99% 69% 75% 83% 99% 98% 99% 0.69
COLLAB 102% 113% 108% 107% 109% 68% 68% 87% 101% 101% 91% 0.84
REDDIT-BINARY 101% 94% 102% 102% 103% 71% 67% 67% 88% 97% 81% 0.94
REDDIT-MULTI-5K 100% 97% 101% 100% 101% 77% 68% 80% 95% 96% 77% 0.83
PATTERN 99% 100% 100% 100% 100% 54% 55% 82% 100% 100% 69% 0.93
SYNTHETICnew 71% 104% 66% 83% 84% 61% 56% 79% 60% 82% 64% 0.91
CLUSTER 55% 55% 100% 95% 97% 60% 60% 70% 100% 100% 62% 0.91
NoNodeFtrs
NodeDeg
LowPass
MidPass
HighPass
NoEdges
FullyConn
Frag-k1
Frag-k2
Frag-k3
FiedlerFrag
original AUROC

Benchmarking Takeaways

- Cora, Pubmed, Citeseer – the MNISTs of Graphs
  - Often demanded by reviewers
  - High variance depending on splits and hyperparameters
  - Testing only on these provides a narrow view of the performance of GNNs
- We don't have a solid understanding what aspects graph benchmarks evaluate
  - The Benchmark Taxonomy is a great start

→ **It is still not very clear what is the best and most fair method to evaluate GNNs**

Thank You So Much!



Any Questions?