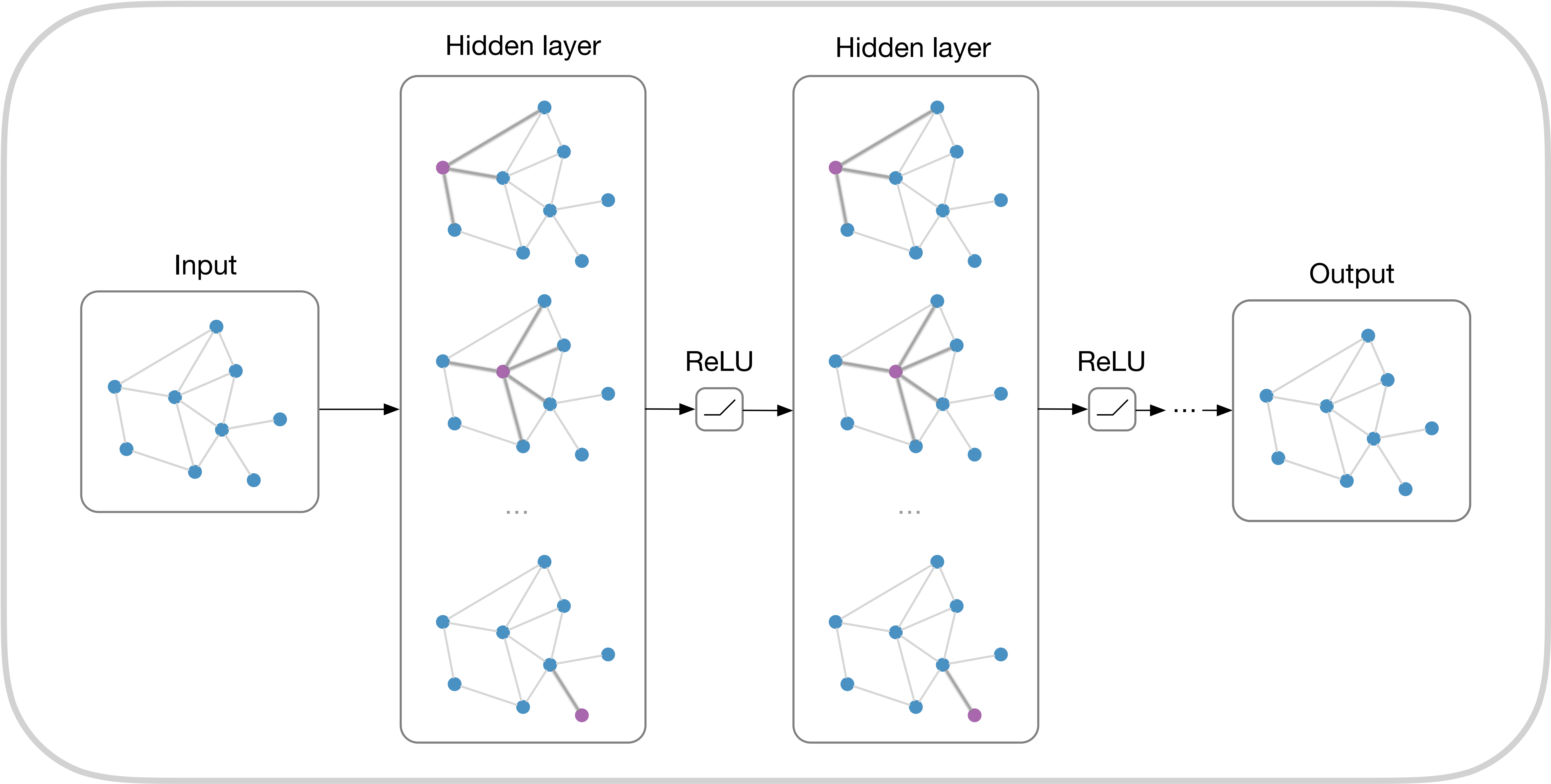
Structured deep models:
Deep learning on graphs and beyond



Thomas Kipf, 25 May 2018

CompBio Seminar, University of Cambridge

Universiteit van Amsterdam

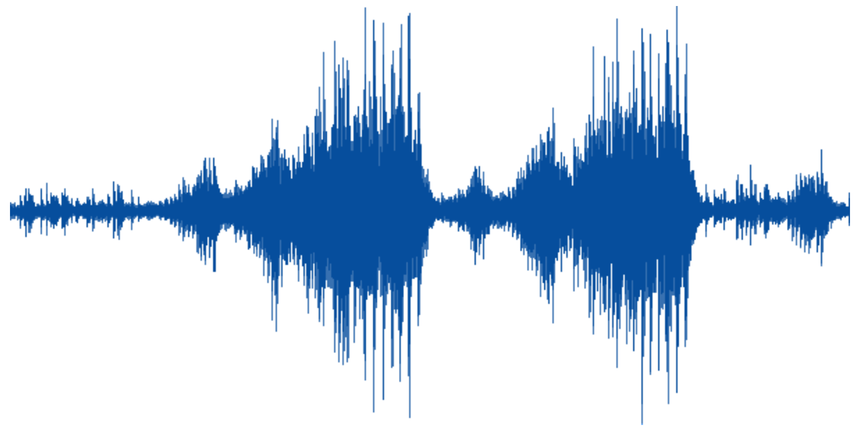
In collaboration with Ethan Fetaya, Rianne van den Berg, Michael Schlichtkrull, Petar Veličković, Ivan Titov, Max Welling, Richard Zemel and others

AMLAB Amsterdam Machine Learning Lab

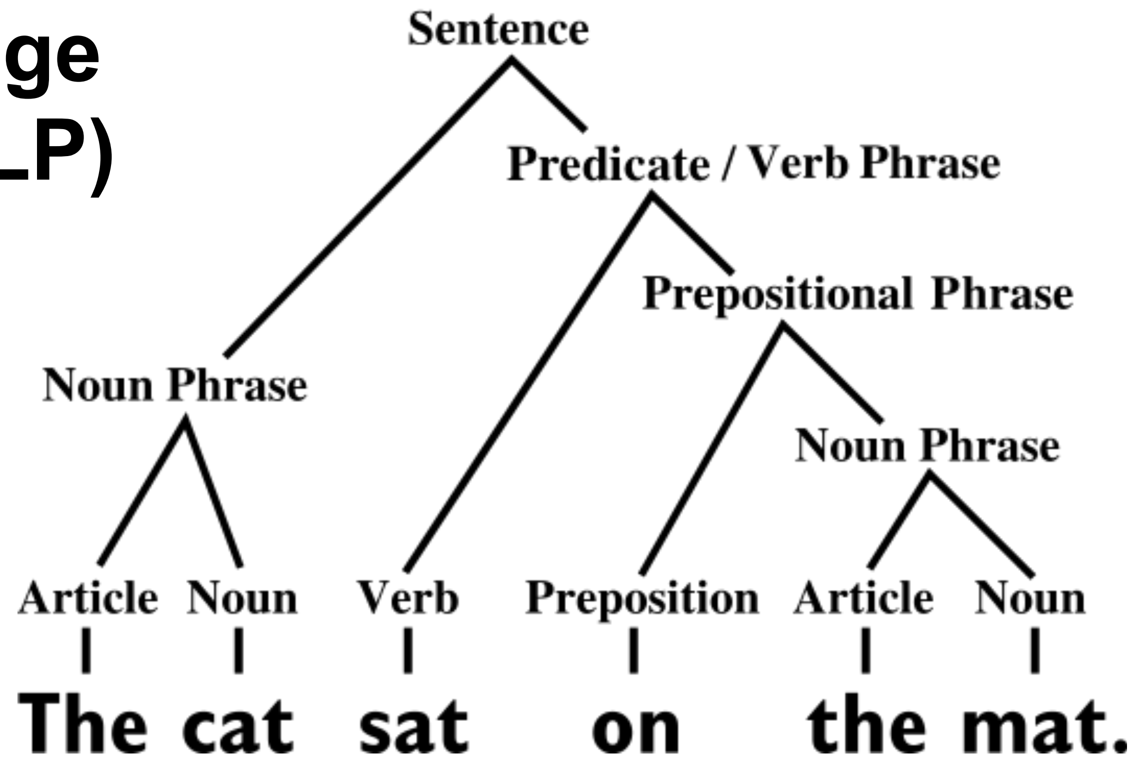
The Deep Learning slide

IMAGENET

Speech data

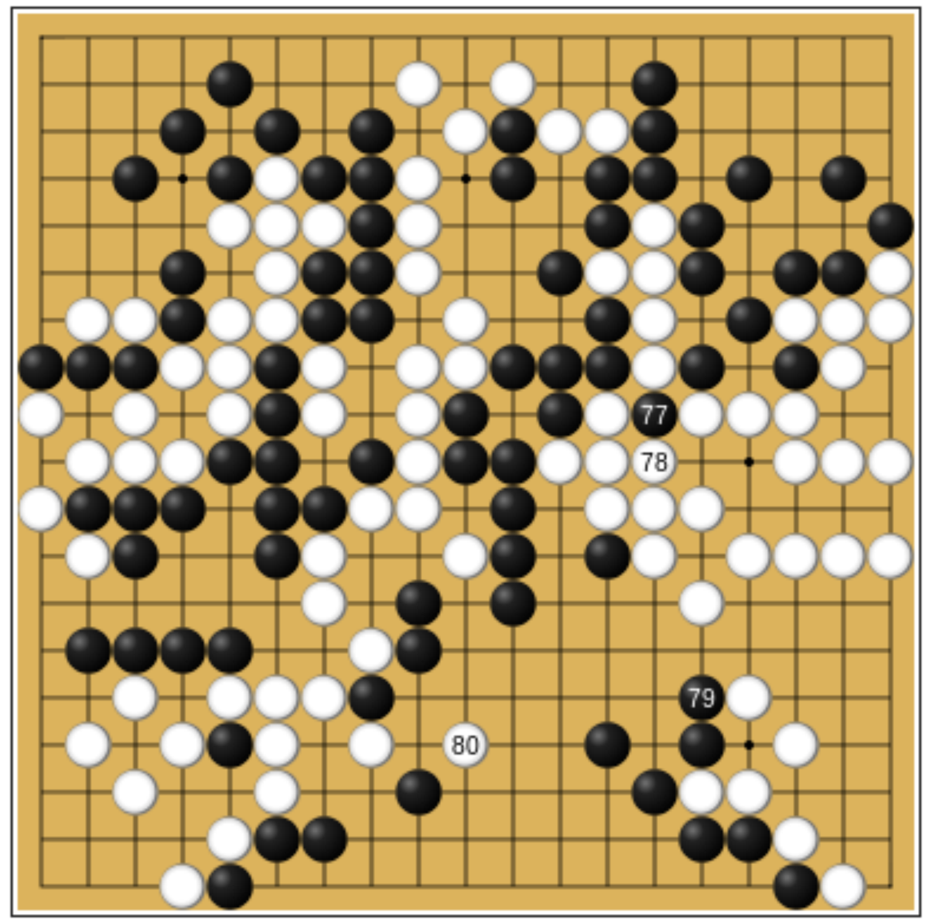


Natural language processing (NLP)



…

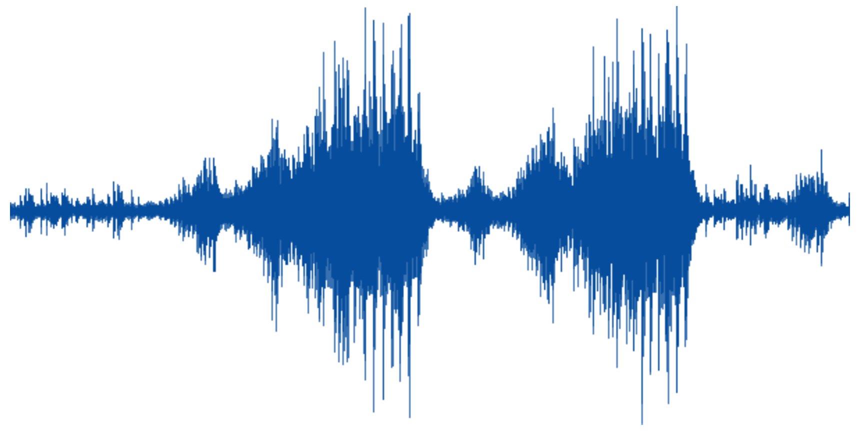
Grid games



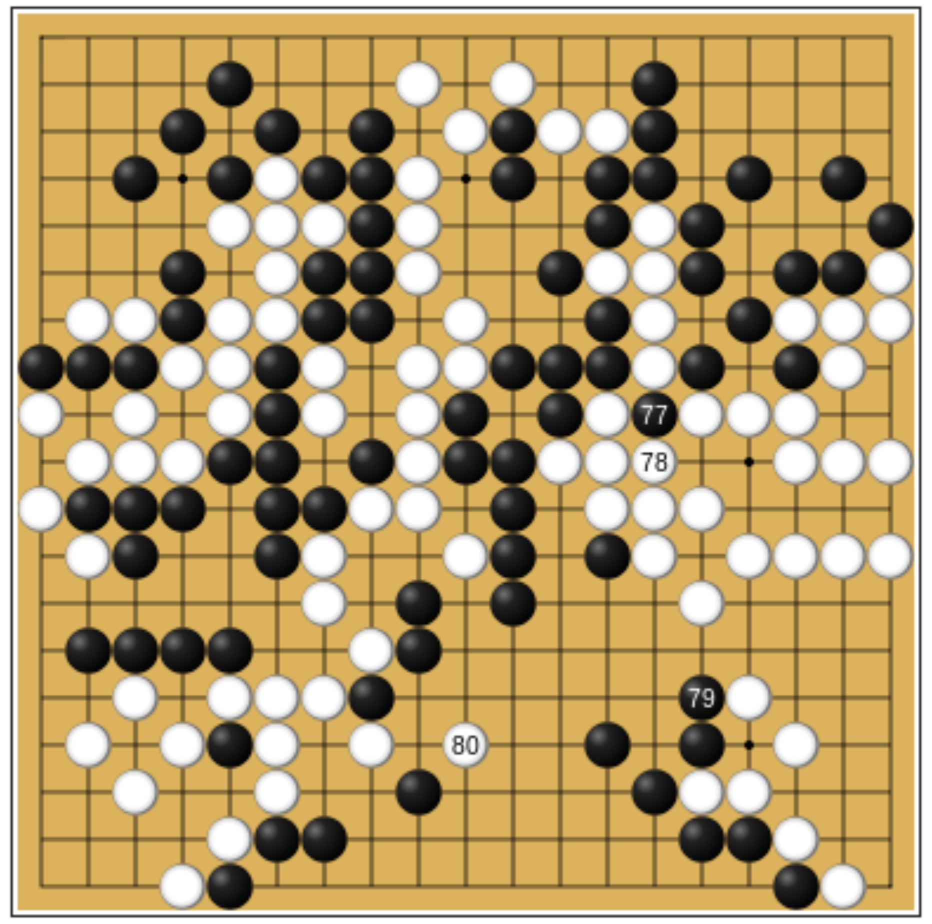
The Deep Learning slide

IMAGENET

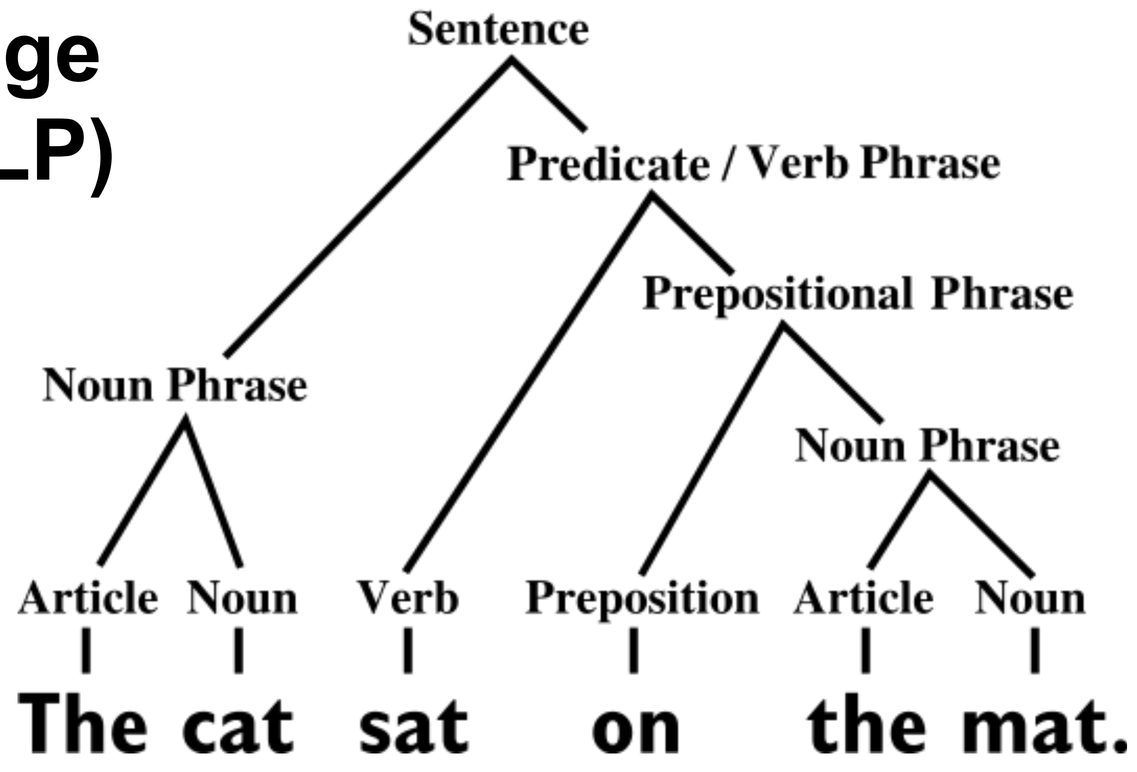
Speech data



Grid games



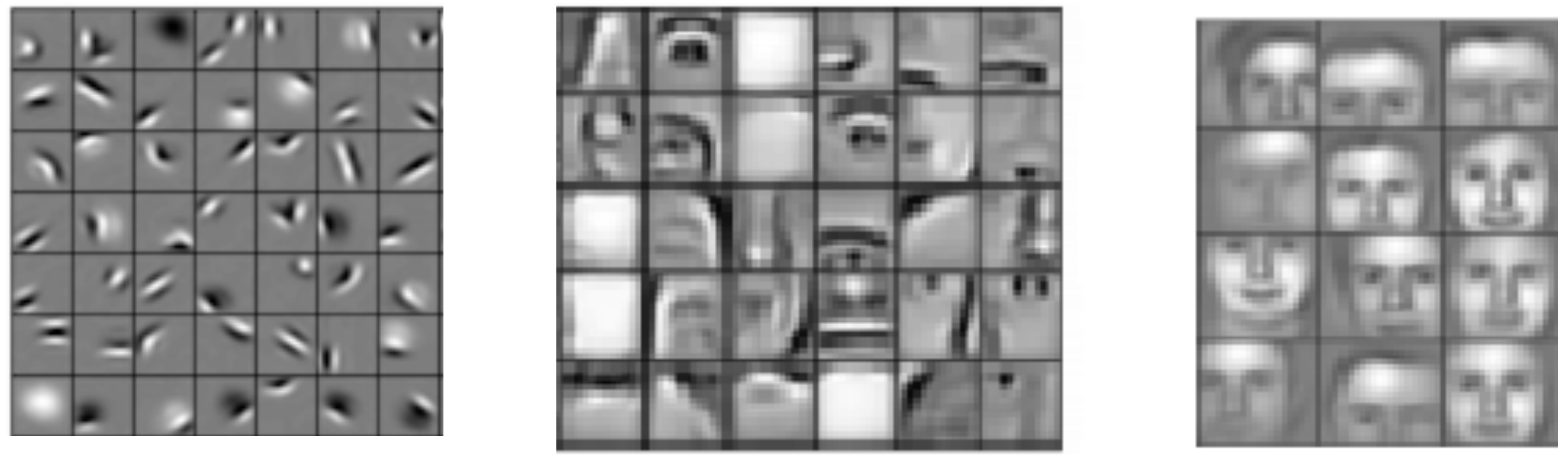
Natural language processing (NLP)



...

Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality



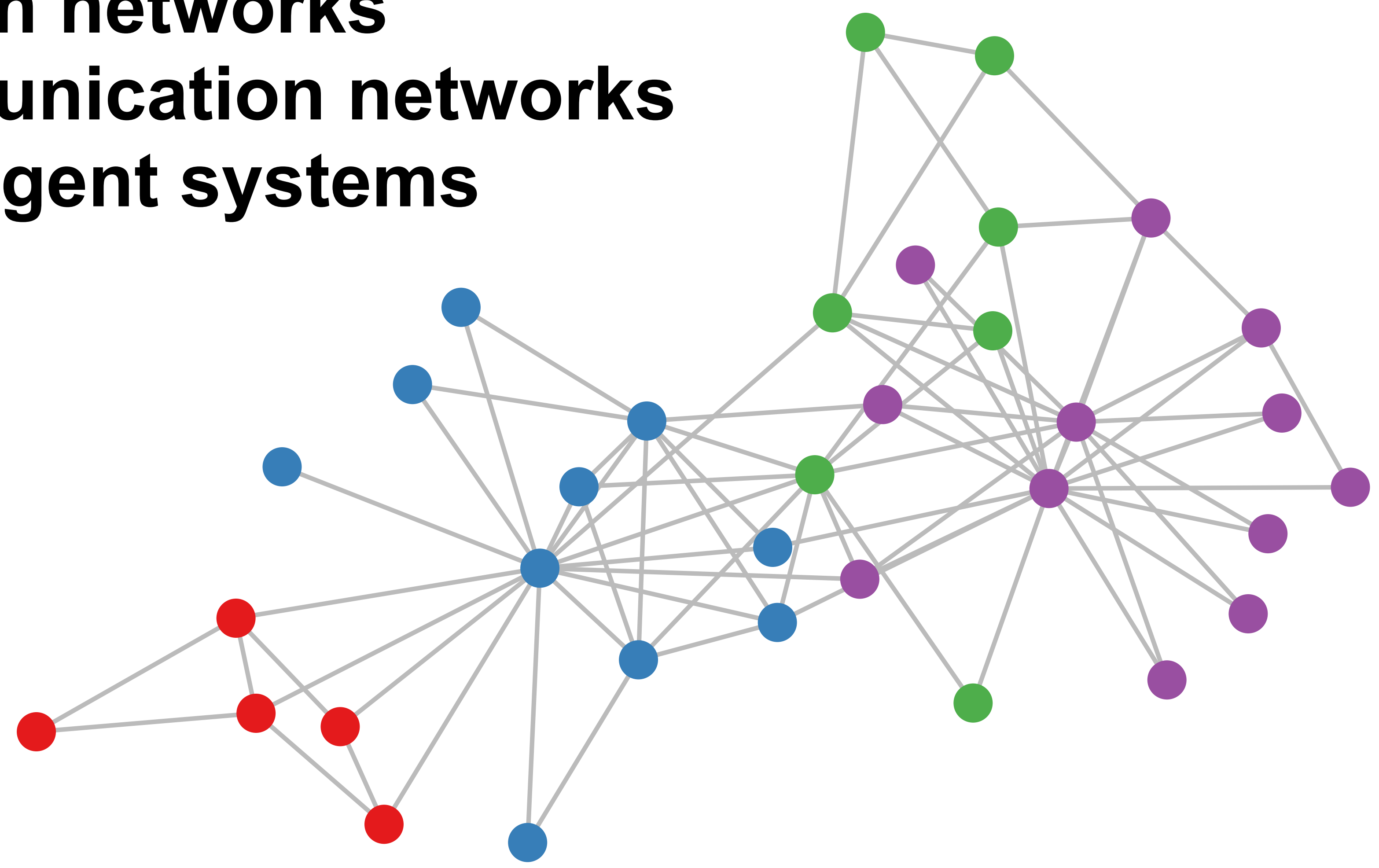
Graph-structured data

A lot of real-world data does not “live” on grids

Graph-structured data

A lot of real-world data does not “live” on grids

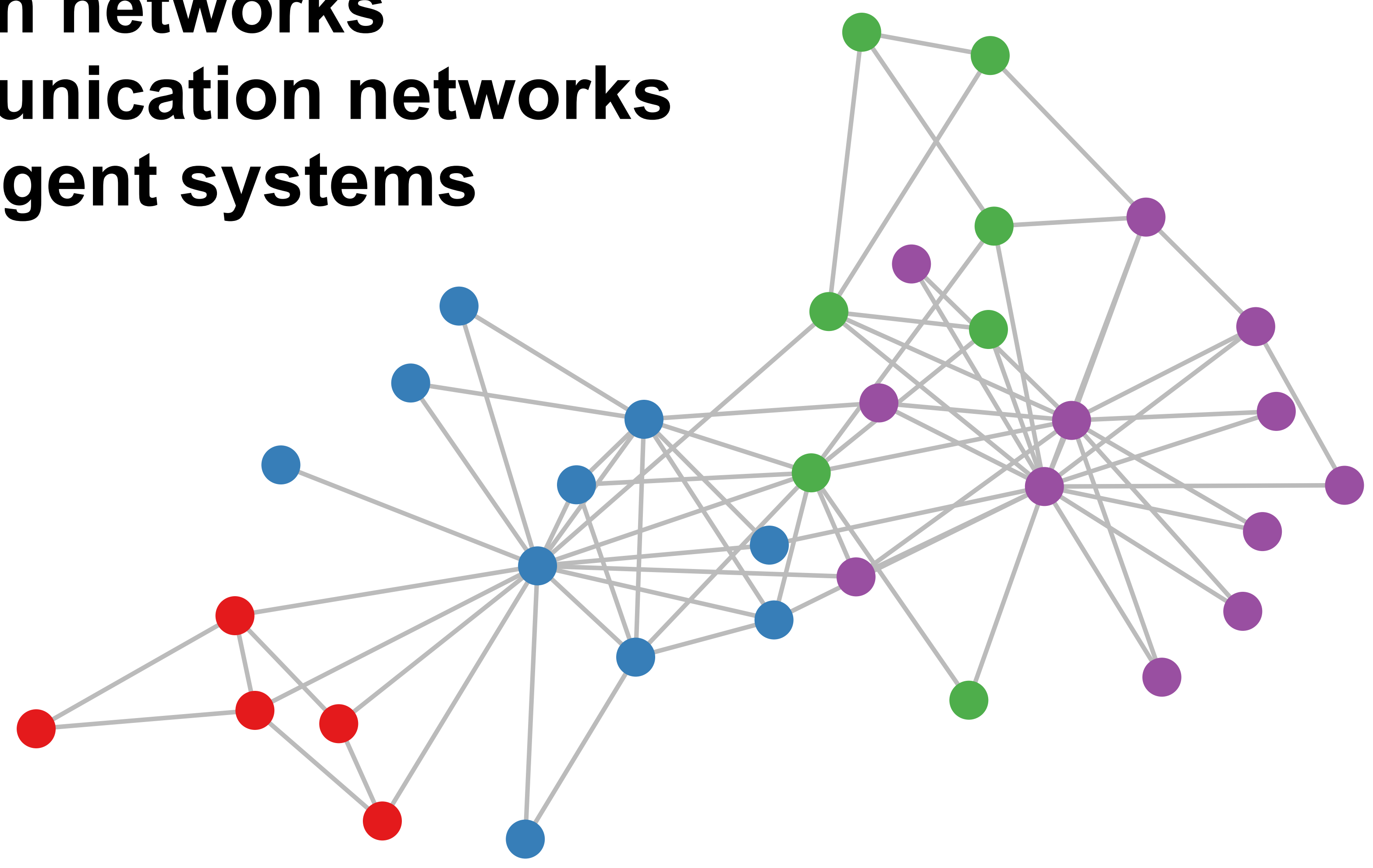
Social networks
Citation networks
Communication networks
Multi-agent systems

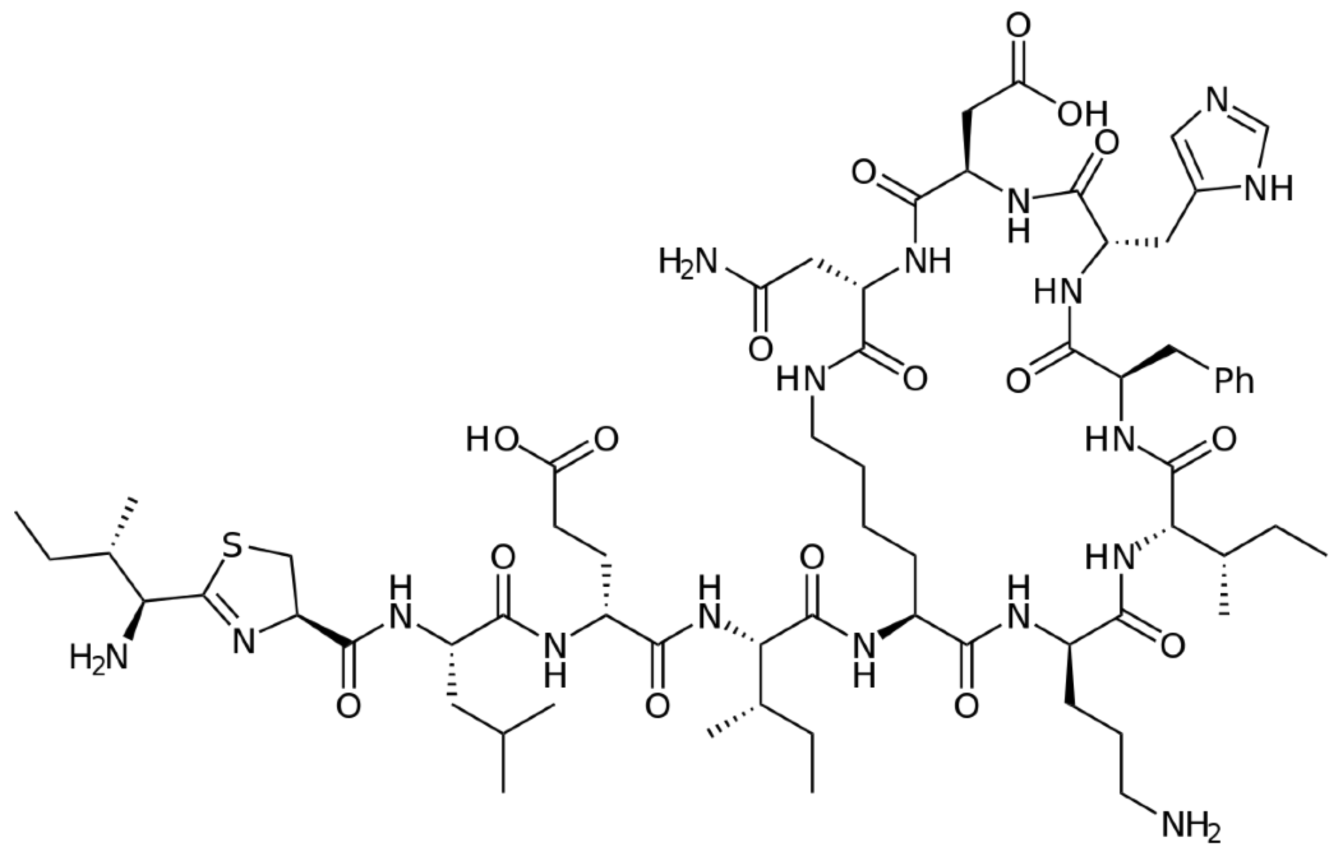


Graph-structured data

A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems



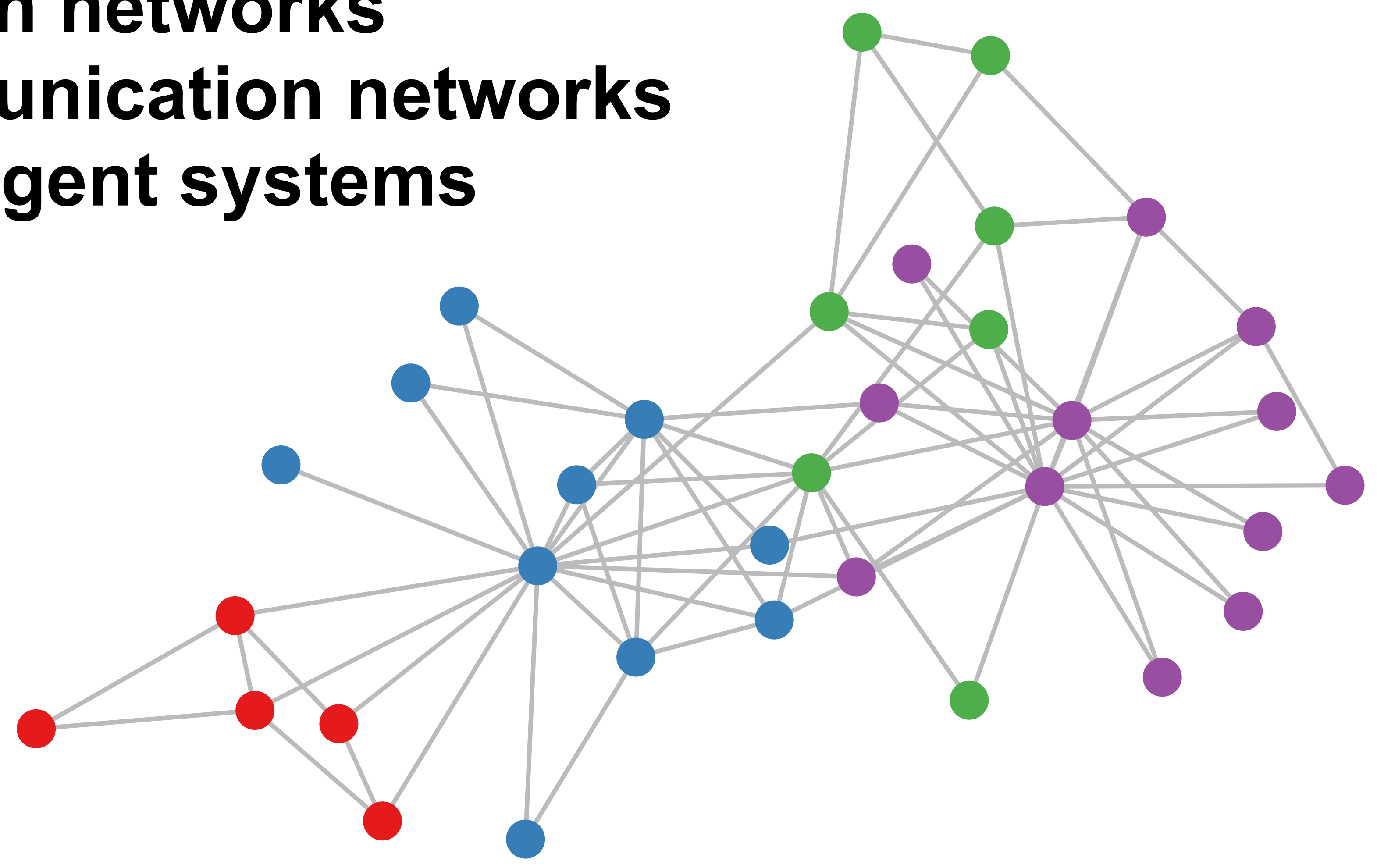


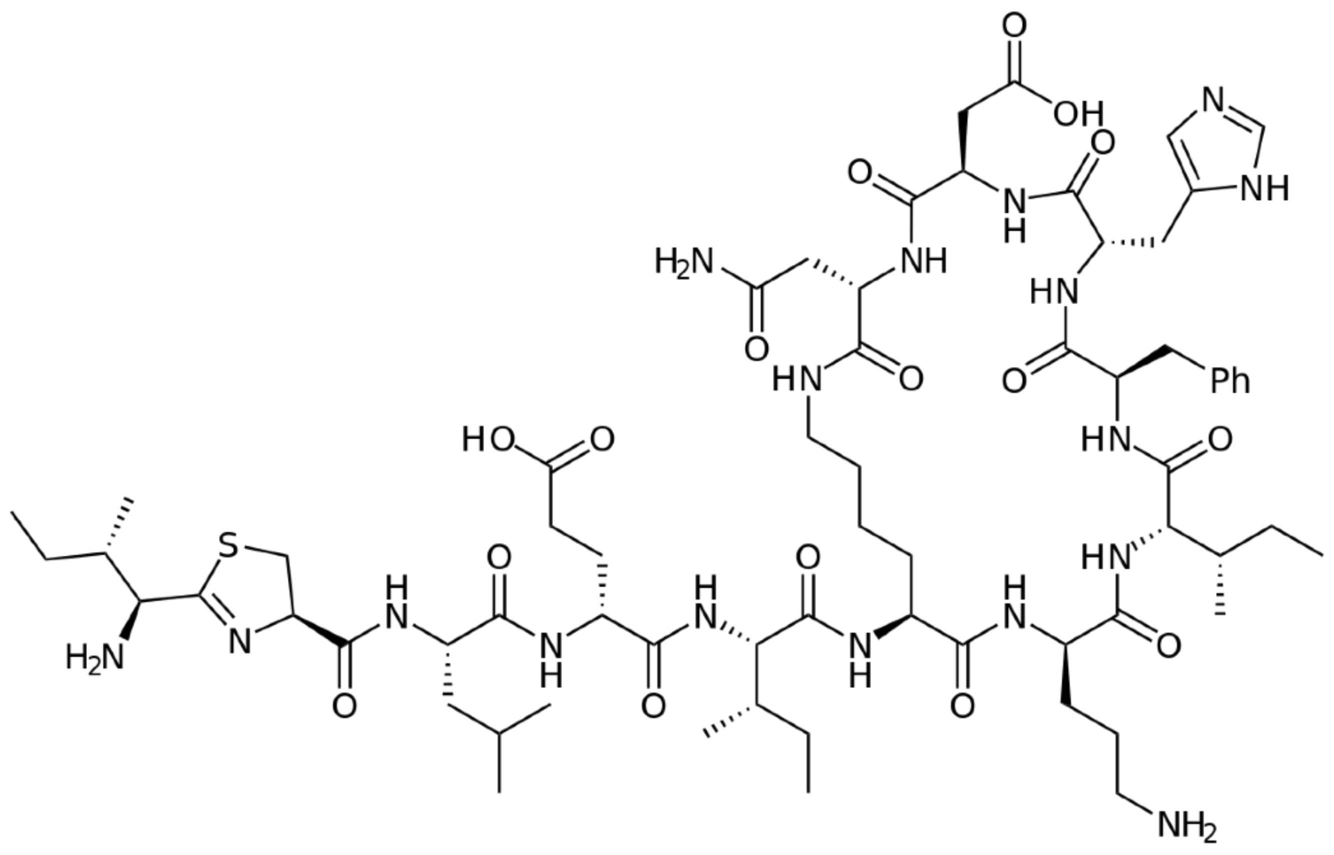
Molecules

Graph-structured data

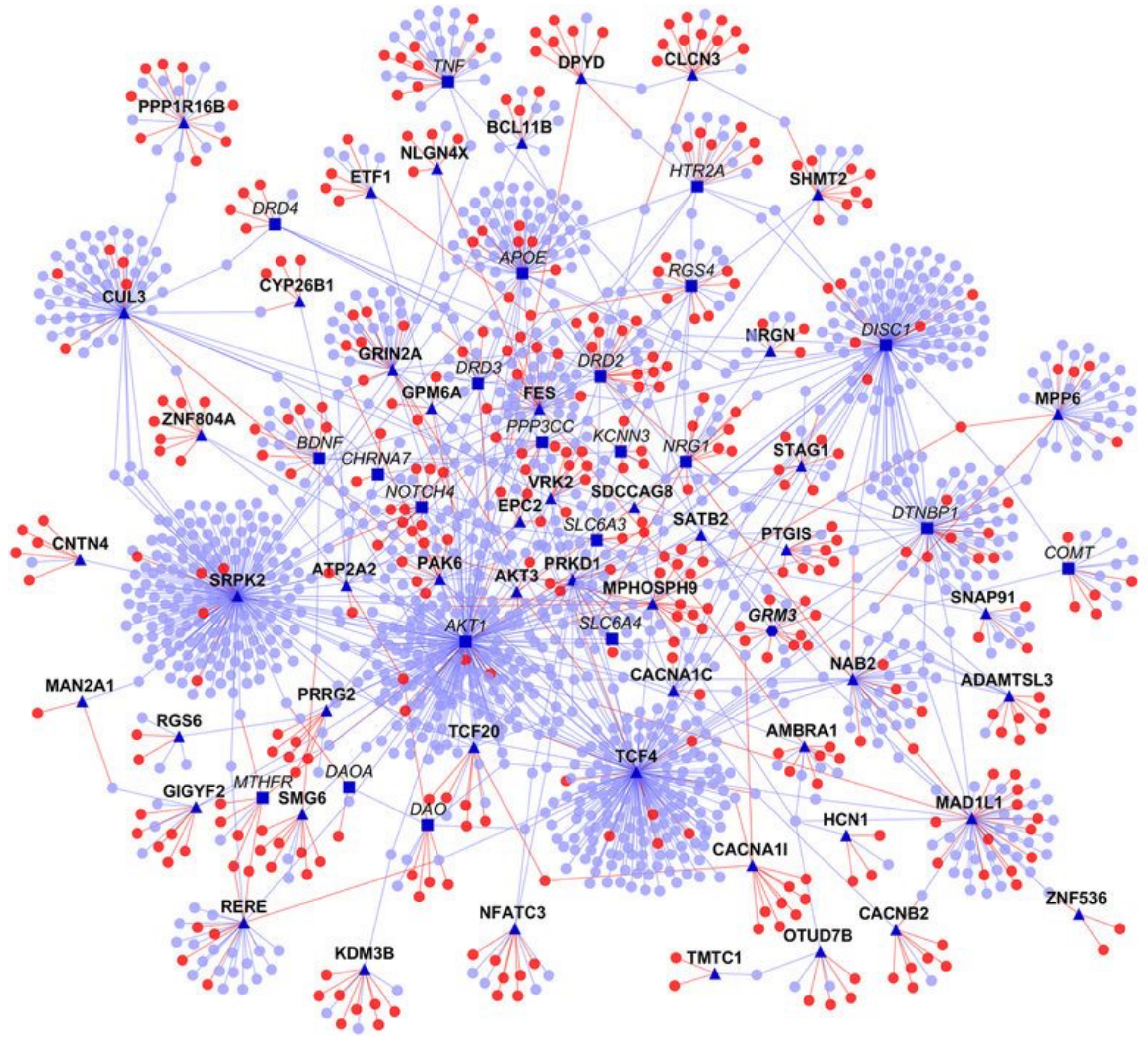
A lot of real-world data does not "live" on grids

Social networks
Citation networks
Communication networks
Multi-agent systems





Molecules

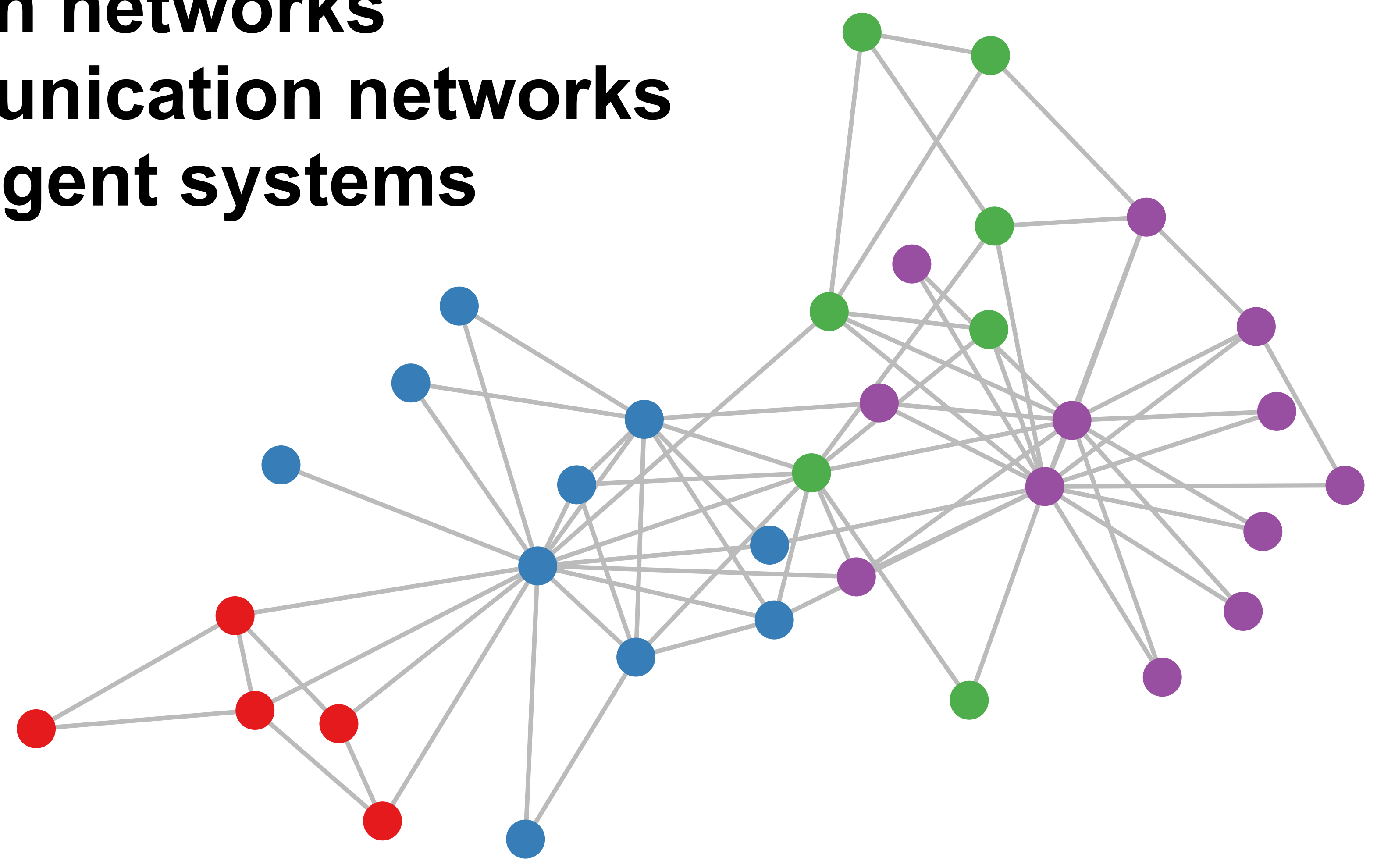


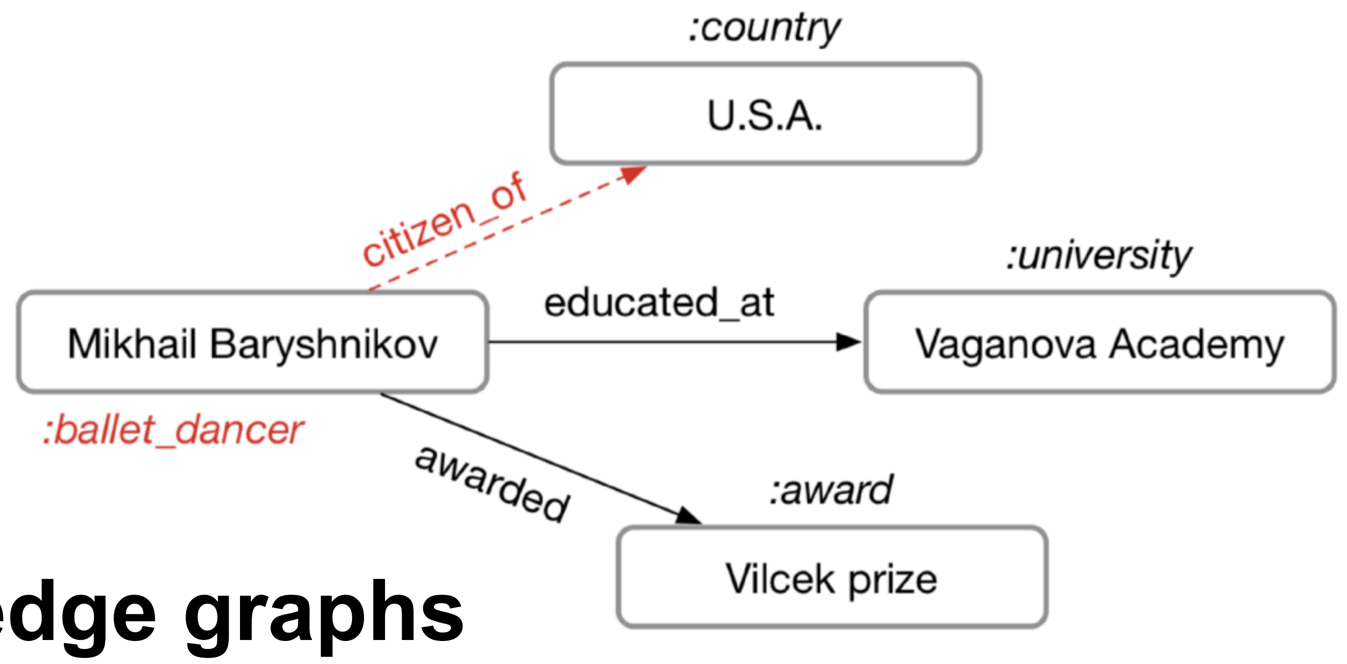
Protein interaction networks

Graph-structured data

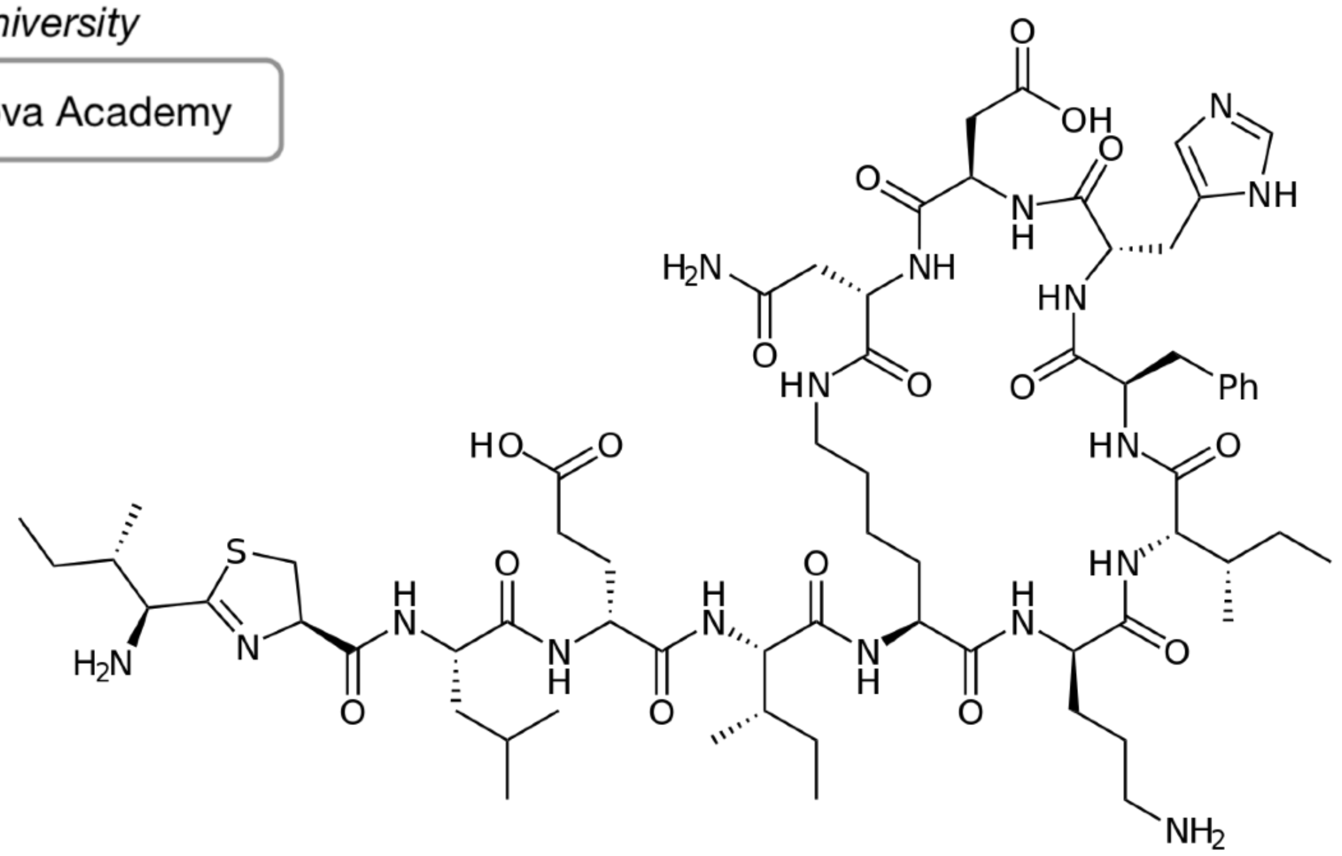
A lot of real-world data does not "live" on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

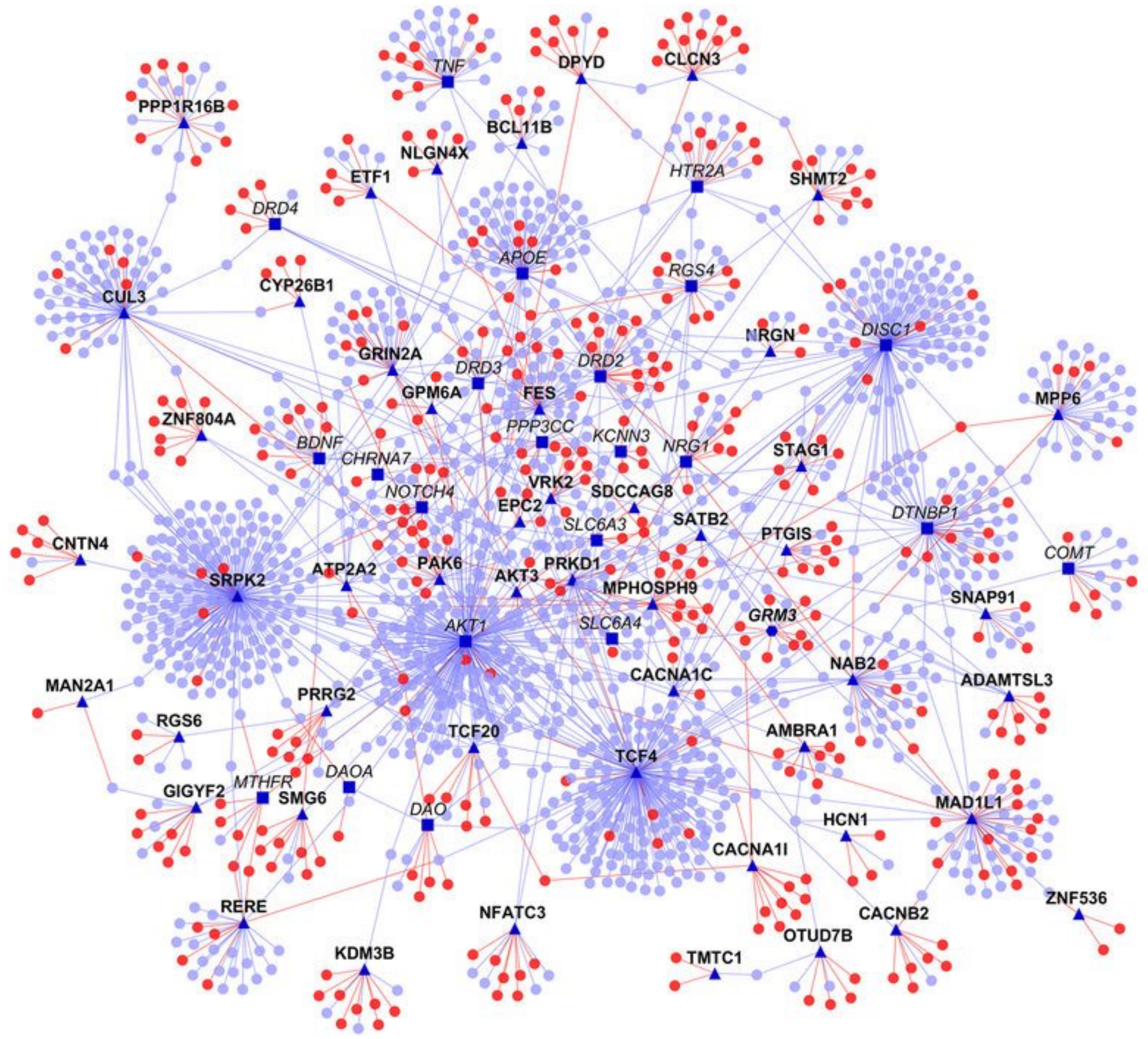




Knowledge graphs



Molecules

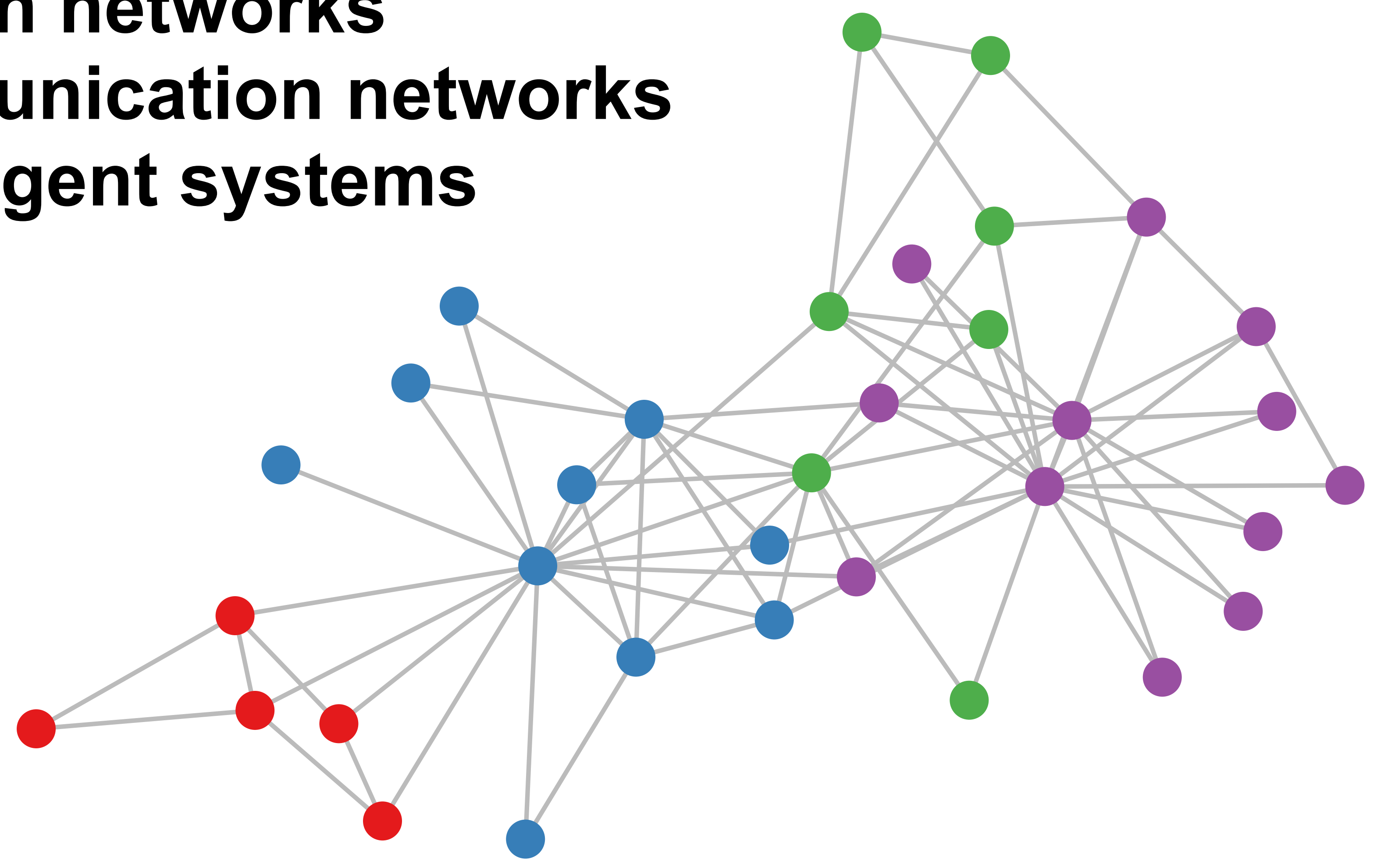


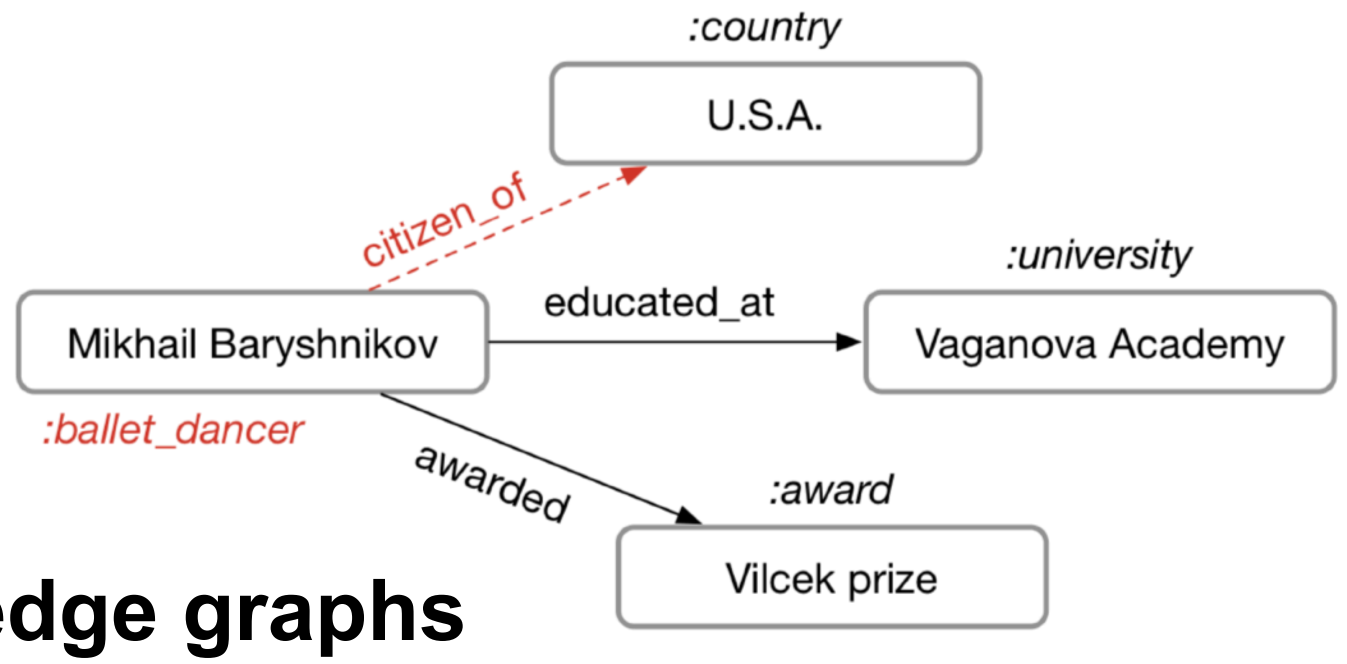
Protein interaction networks

Graph-structured data

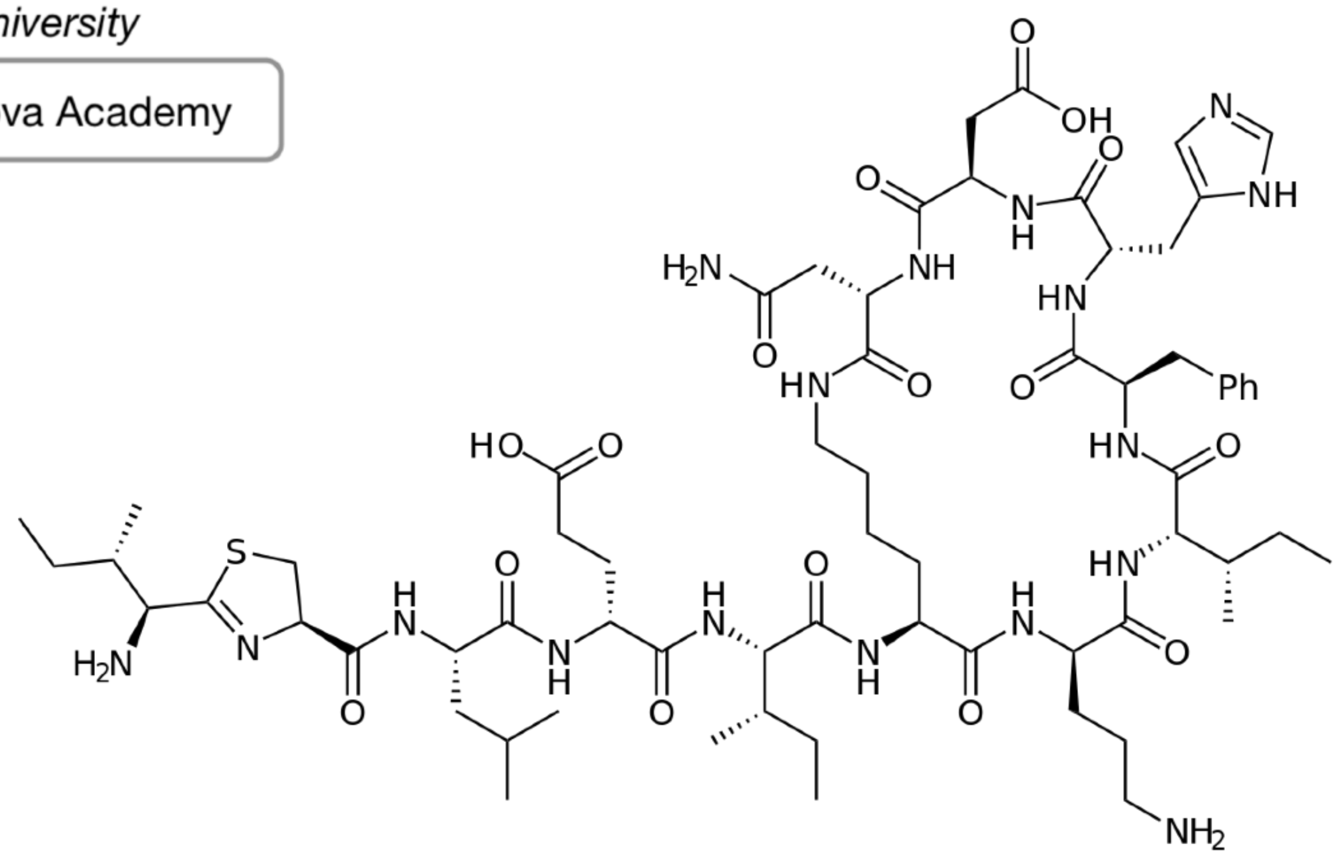
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

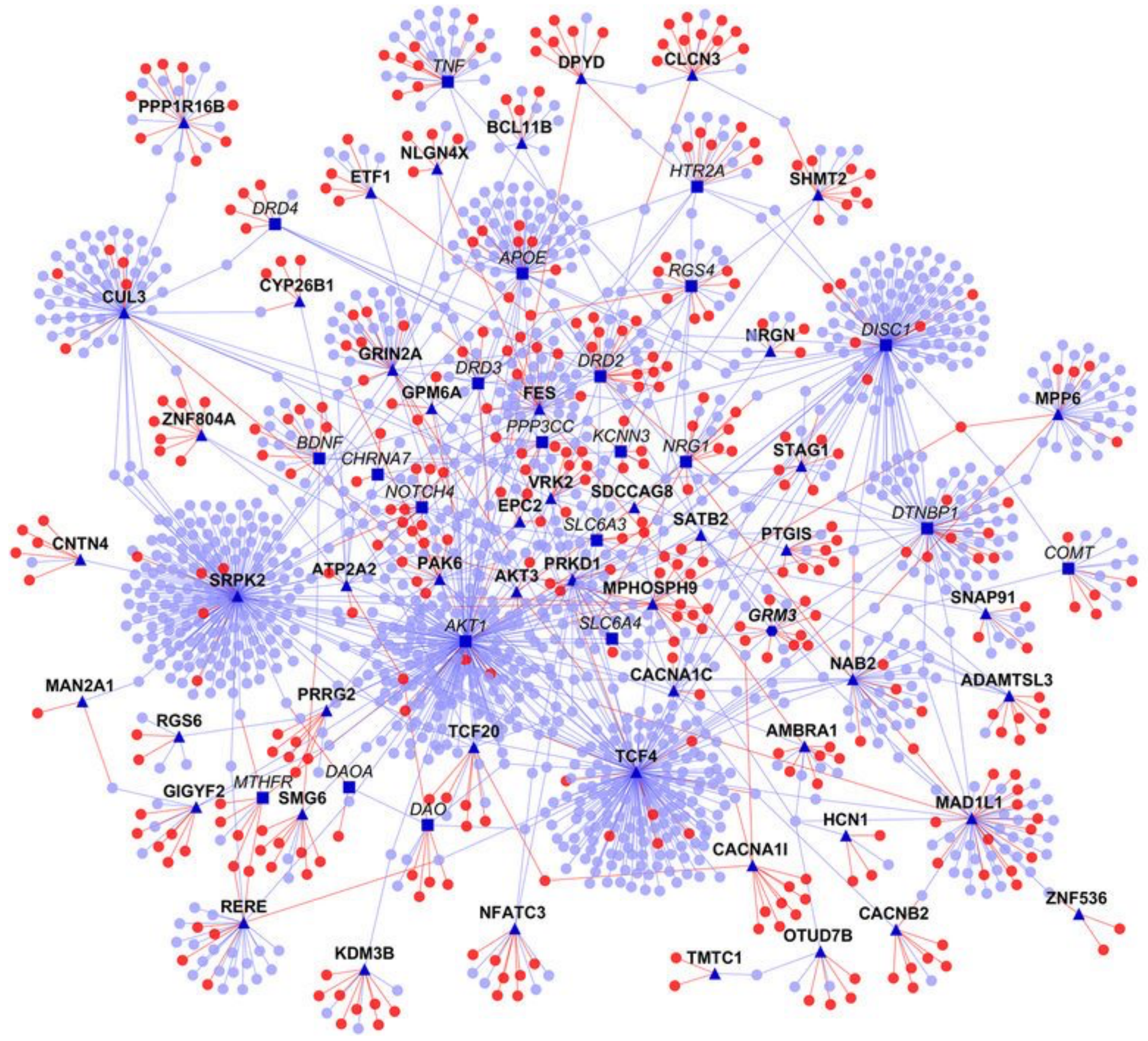




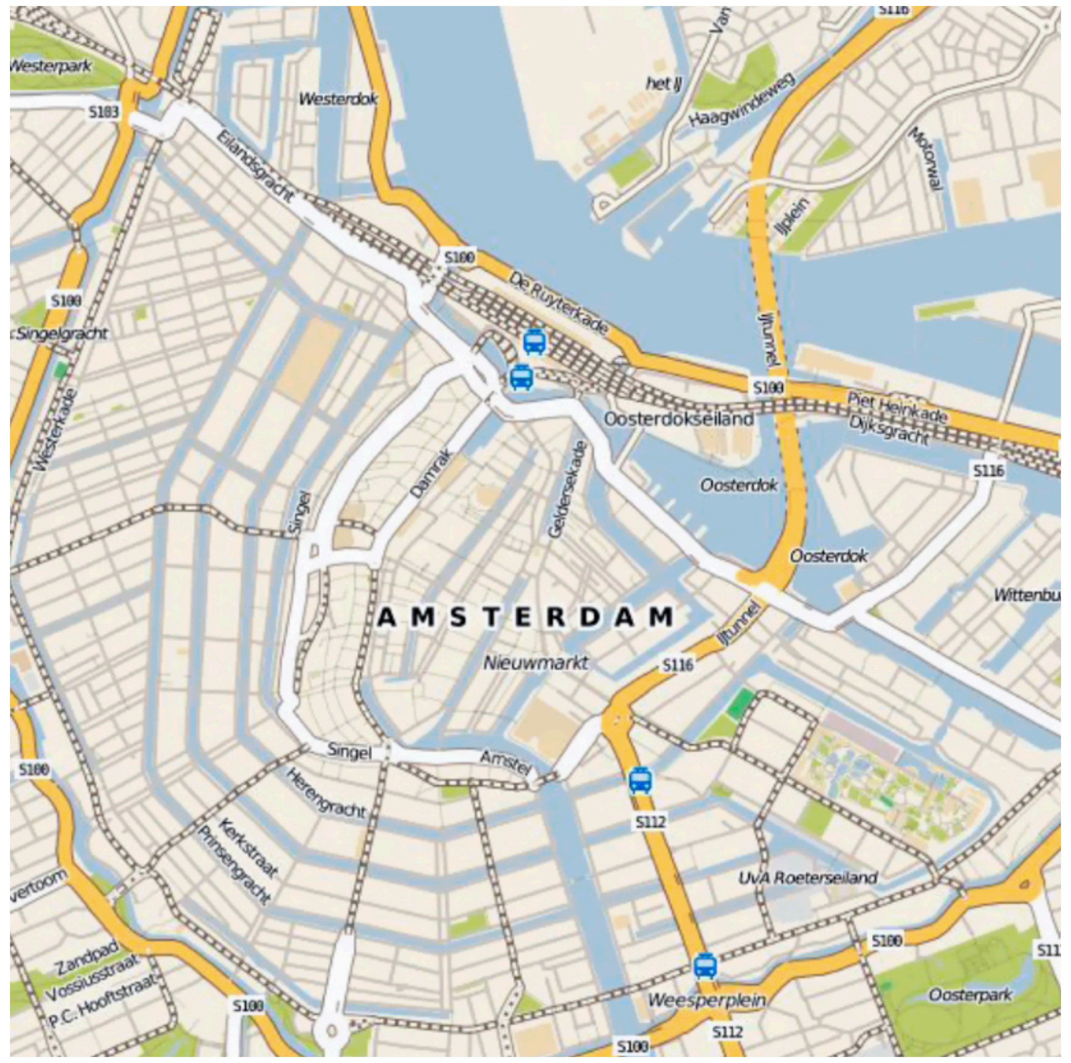
Knowledge graphs



Molecules



Protein interaction networks

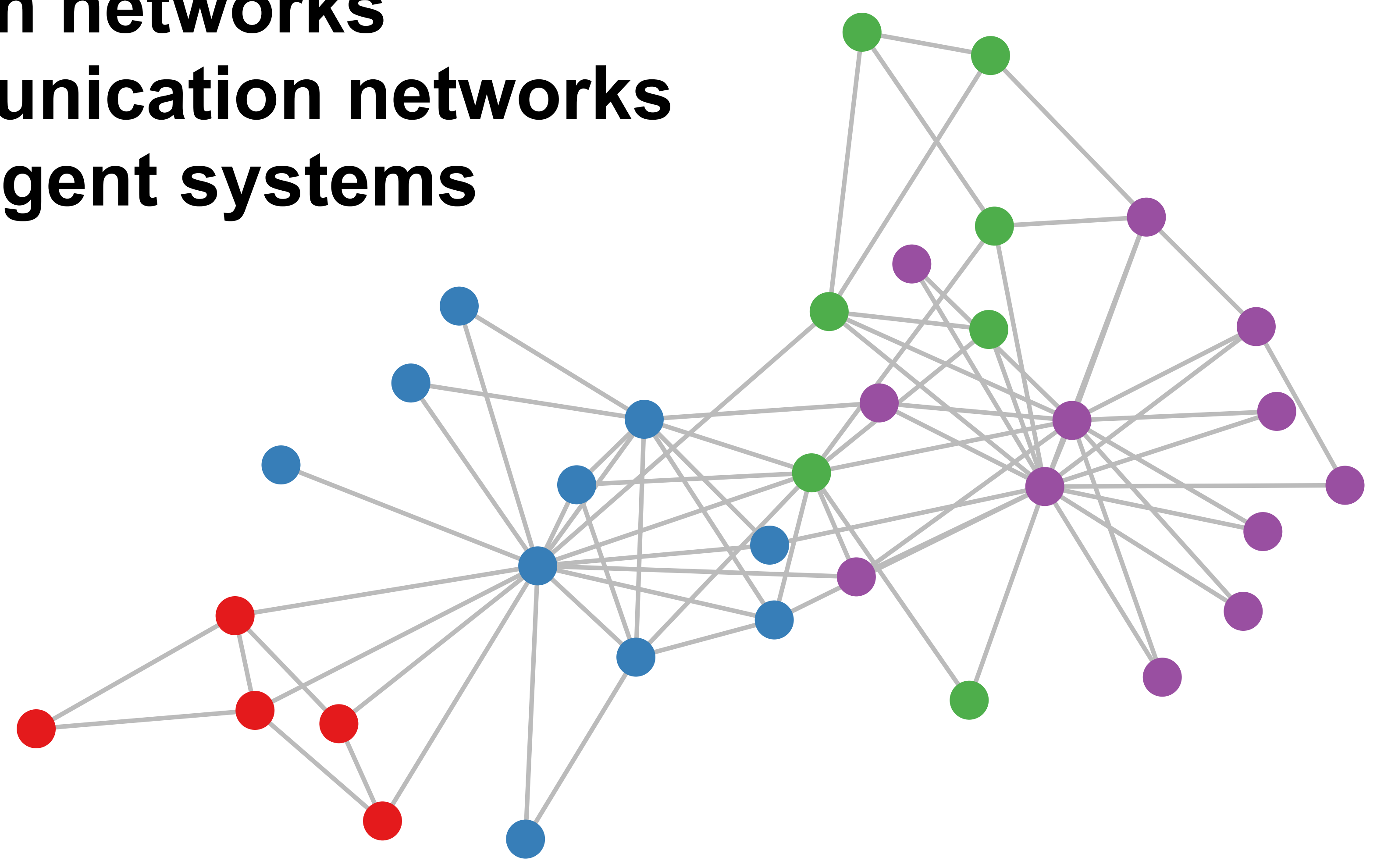


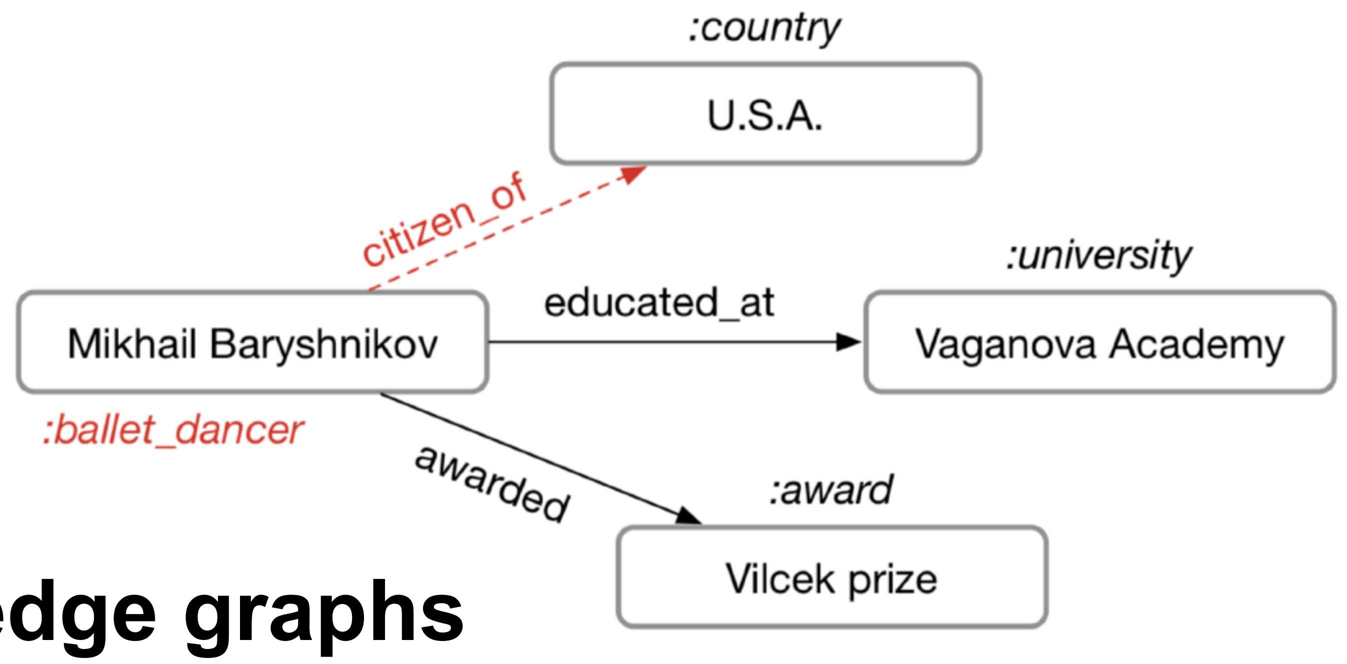
Road maps

Graph-structured data

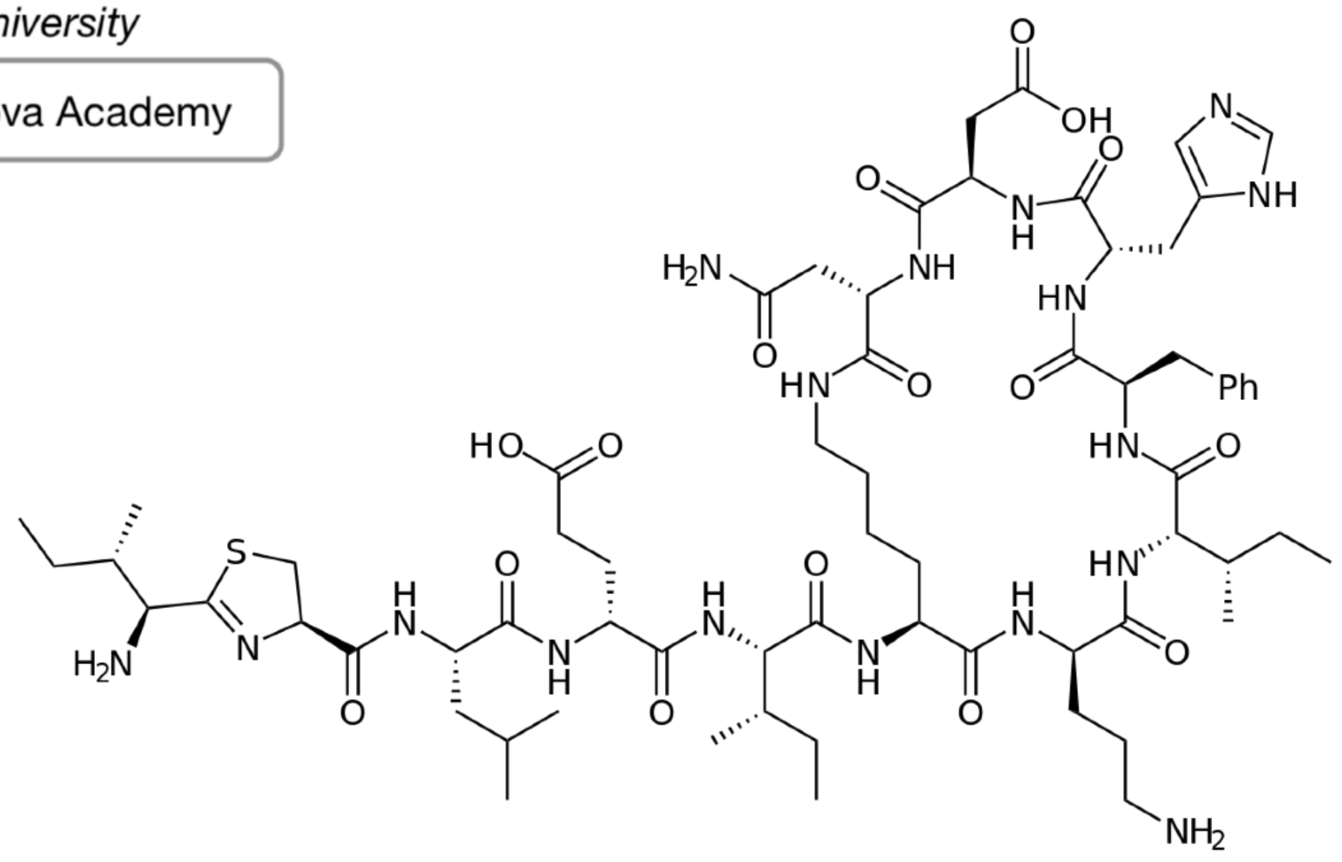
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

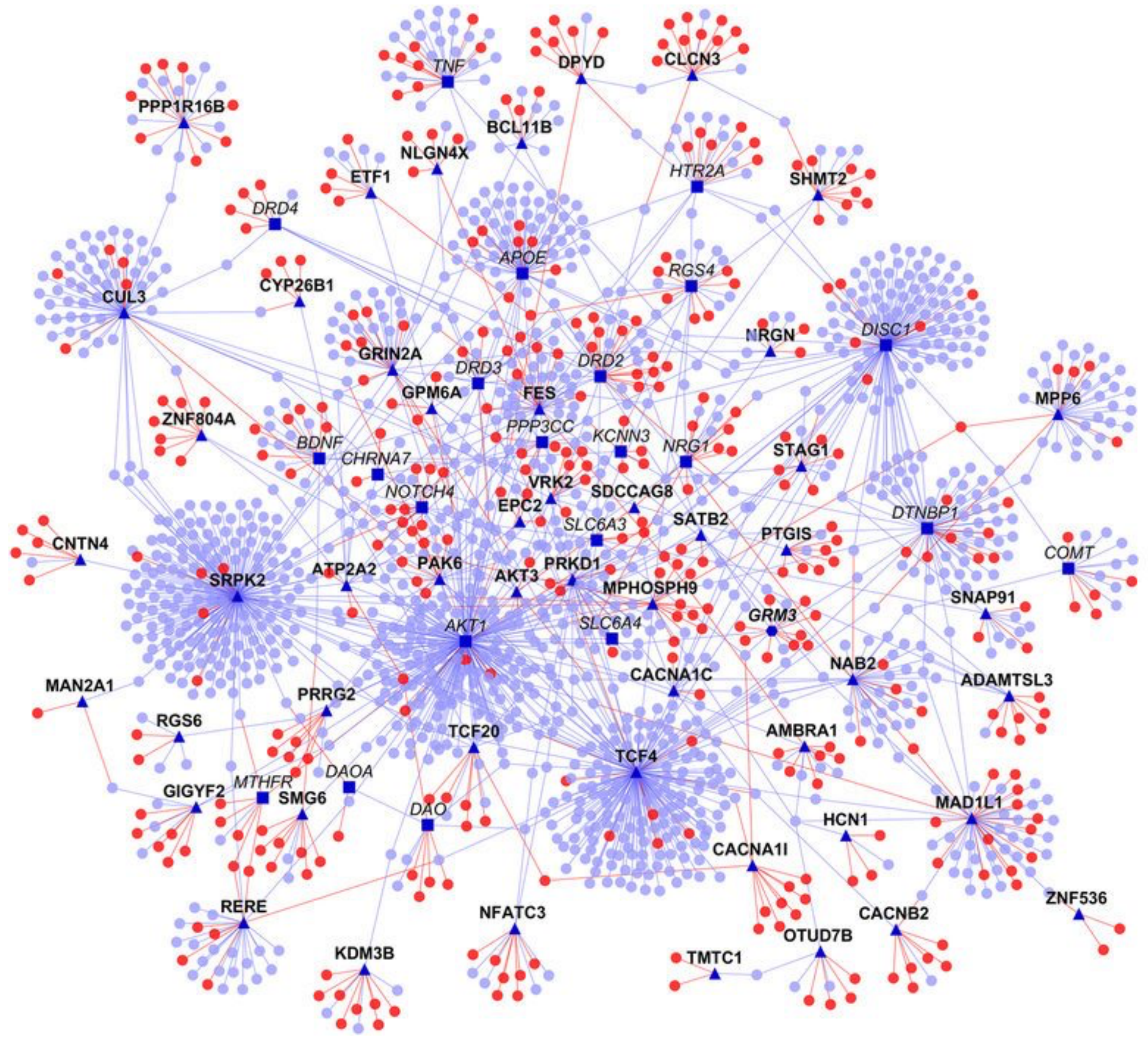




Knowledge graphs

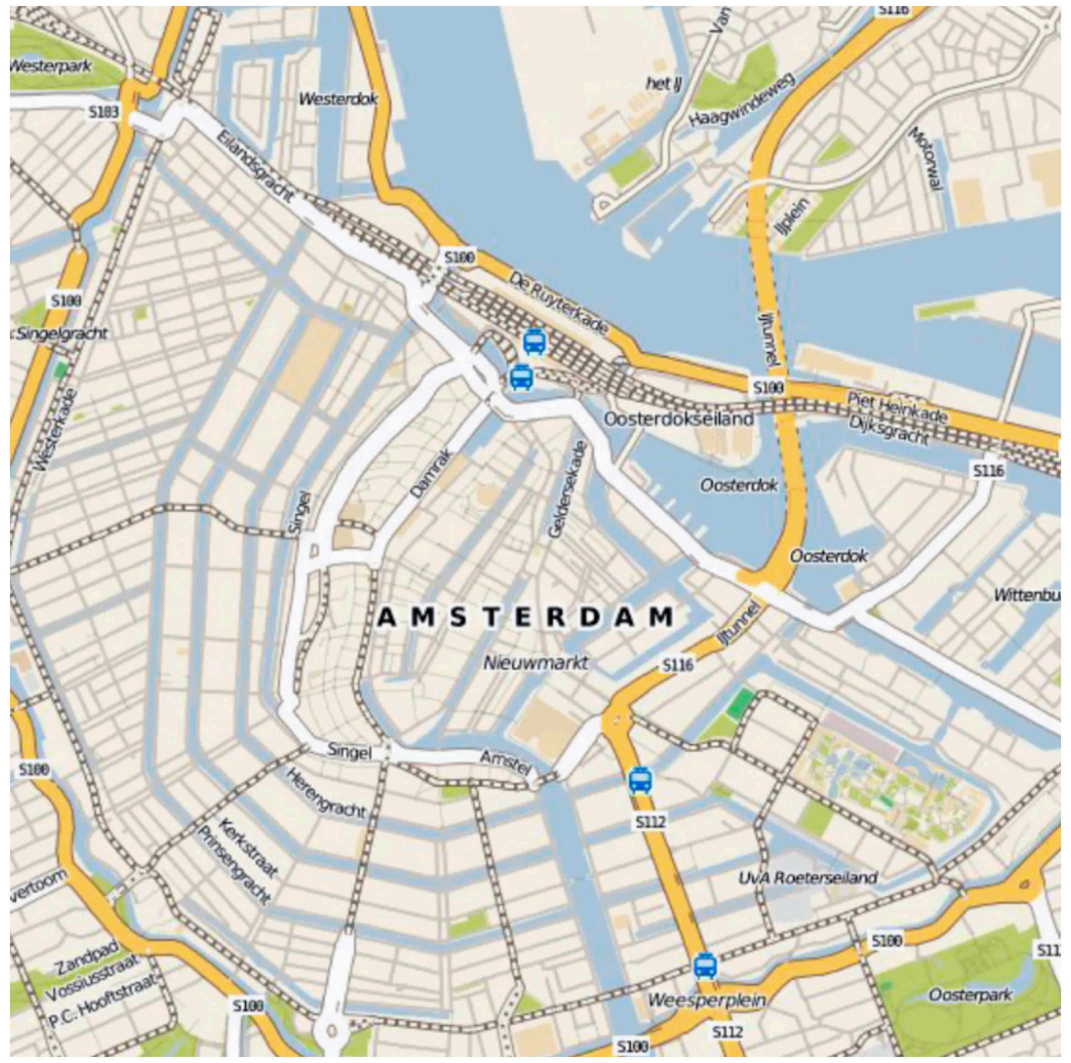


Molecules



Protein interaction networks

Standard deep learning architectures like CNNs and RNNs don’t work here!

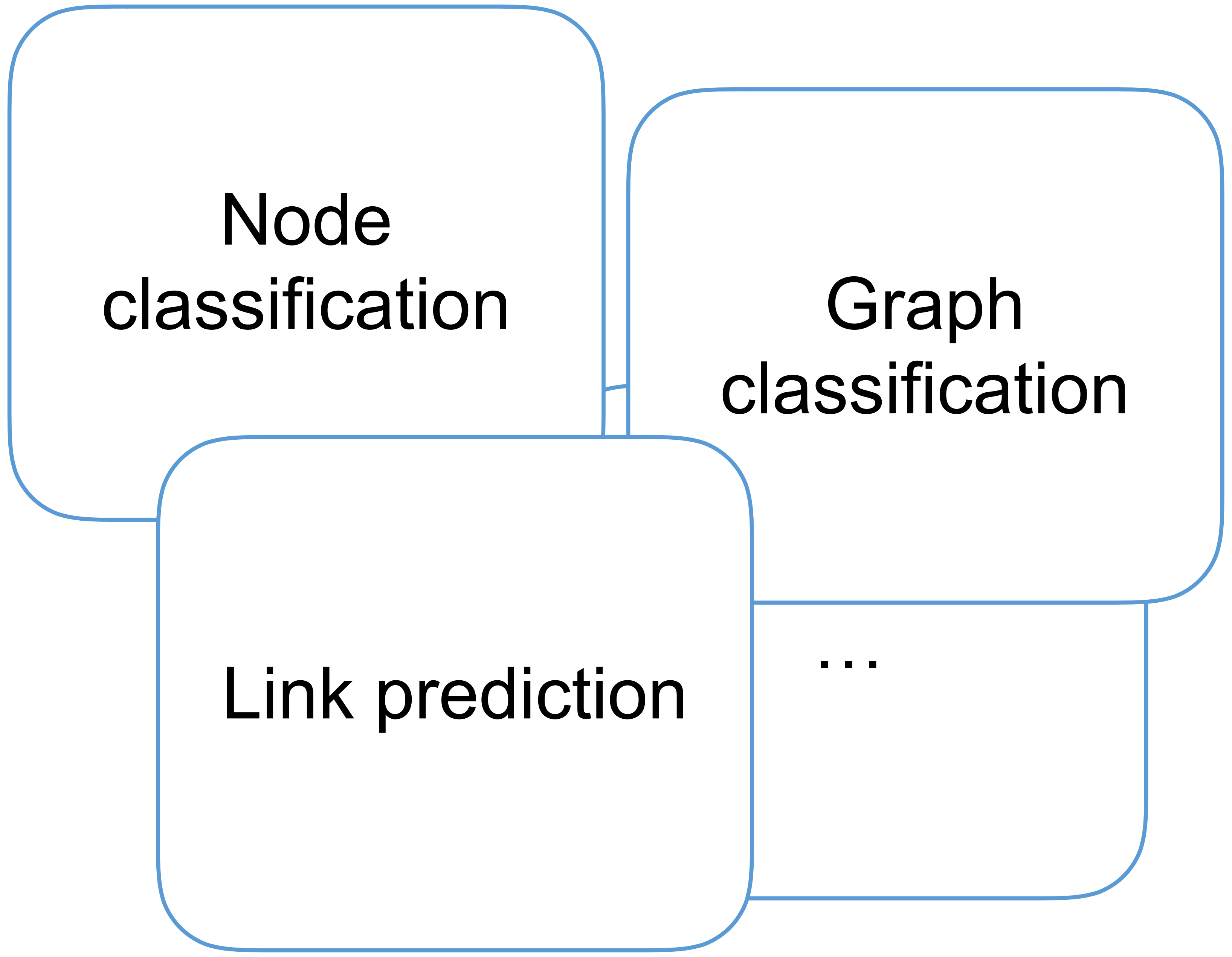


Road maps

Talk overview

1) Graph neural nets (GNNs):
Introduction & some history

2) GNNs for “classical” network problems



Talk overview

1) Graph neural nets (GNNs):
Introduction & some history

2) GNNs for "classical" network problems

Node classification

Graph classification

Link prediction

…

3) Emerging research directions

Latent graph inference

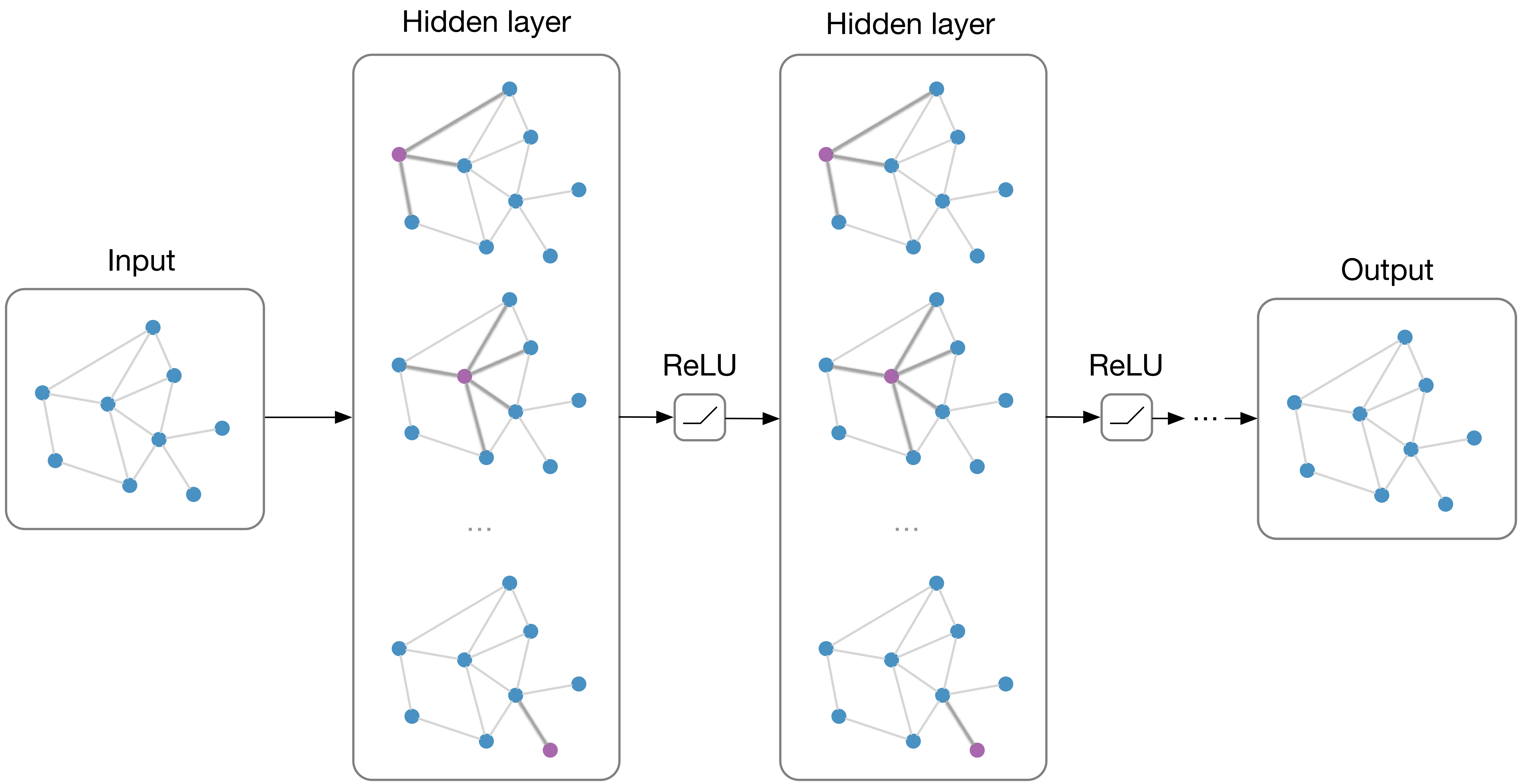
Generative models of graphs

(Potential) applications:

- Interacting systems (physics/multi-agent),
- Causal inference
- Program induction,
- Chemical synthesis,

Graph Neural Networks (GNNs)

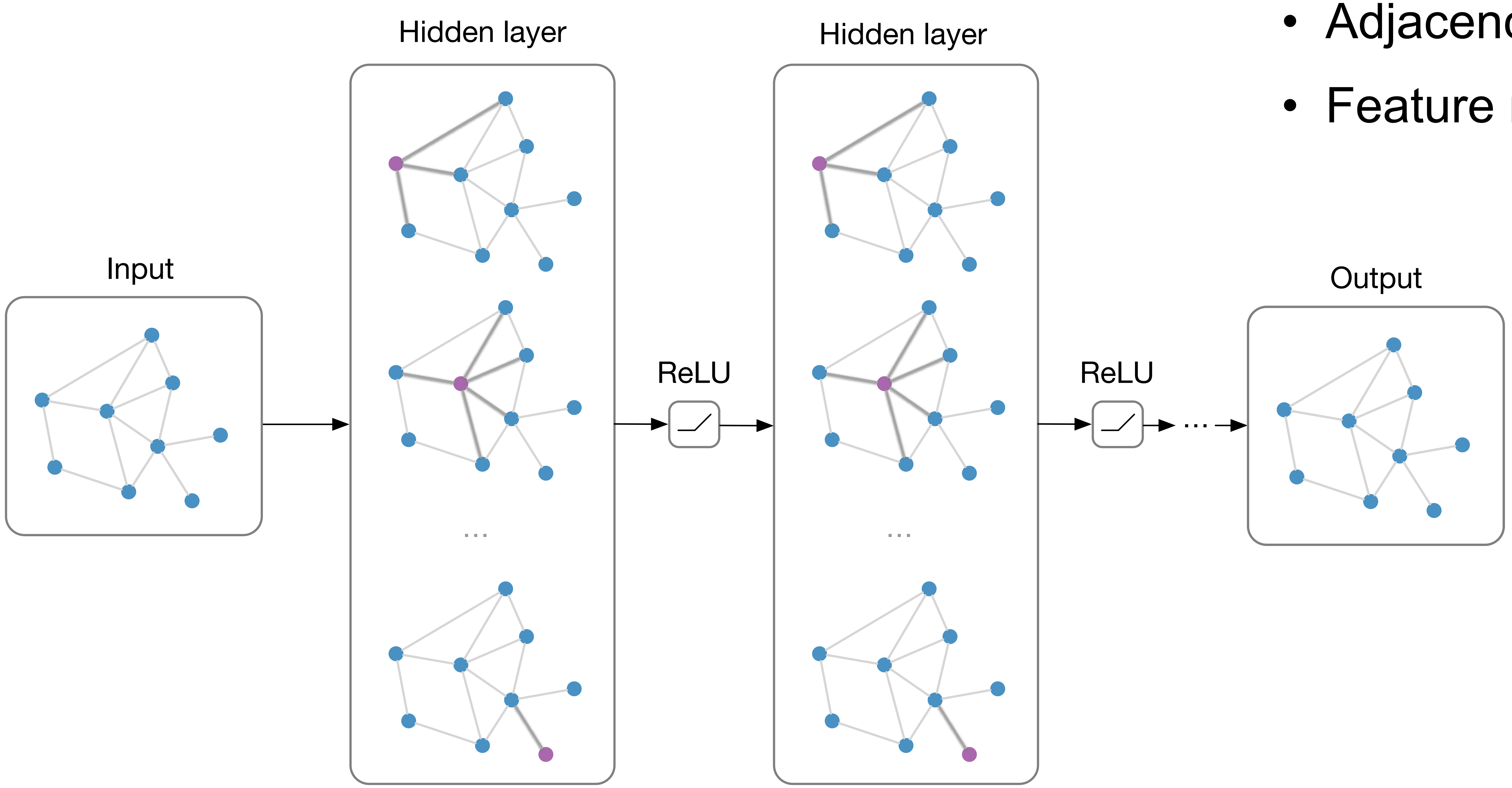
The bigger picture:



Main idea: Pass messages between pairs of nodes & agglomerate

Graph Neural Networks (GNNs)

The bigger picture:



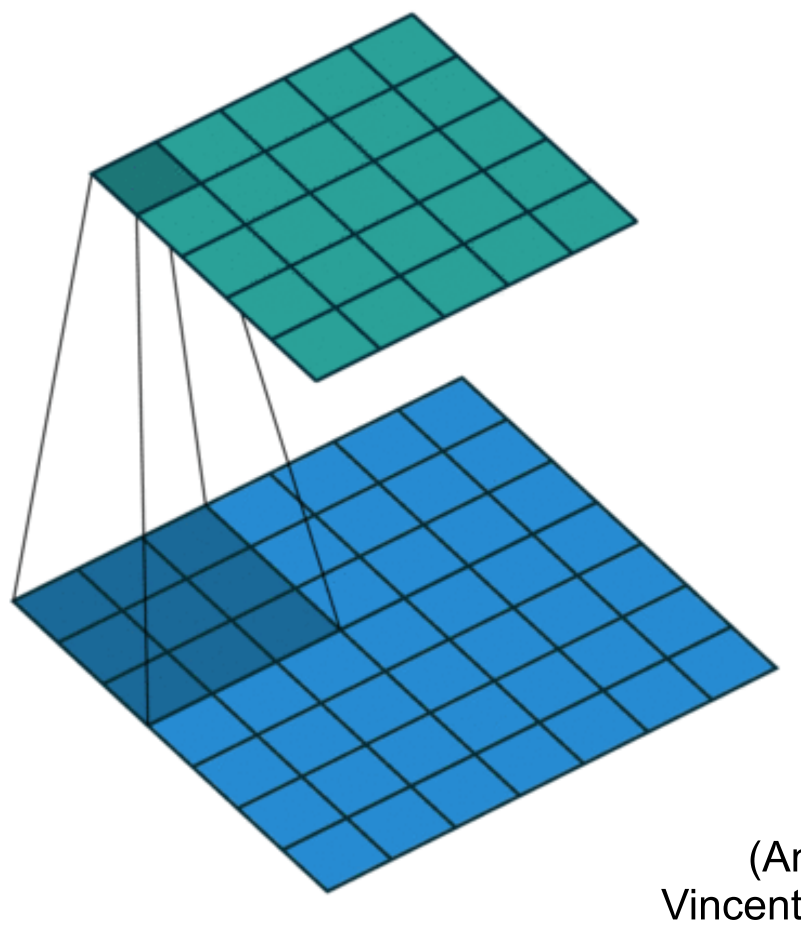
Notation: $\mathcal{G} = (\mathbf{A}, \mathbf{X})$

- Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

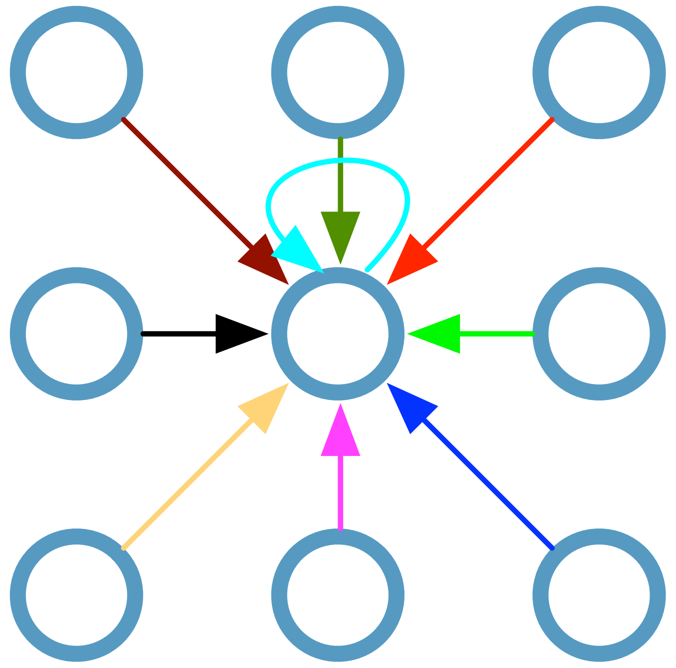
Main idea: Pass messages between pairs of nodes & agglomerate

Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:

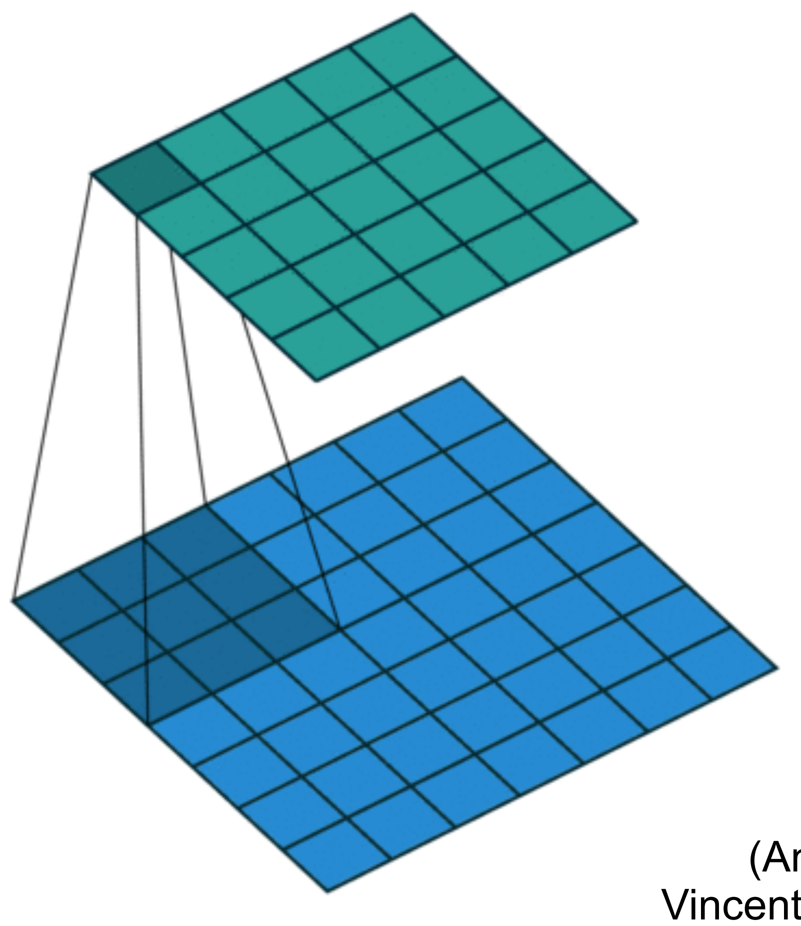


(Animation by Vincent Dumoulin)

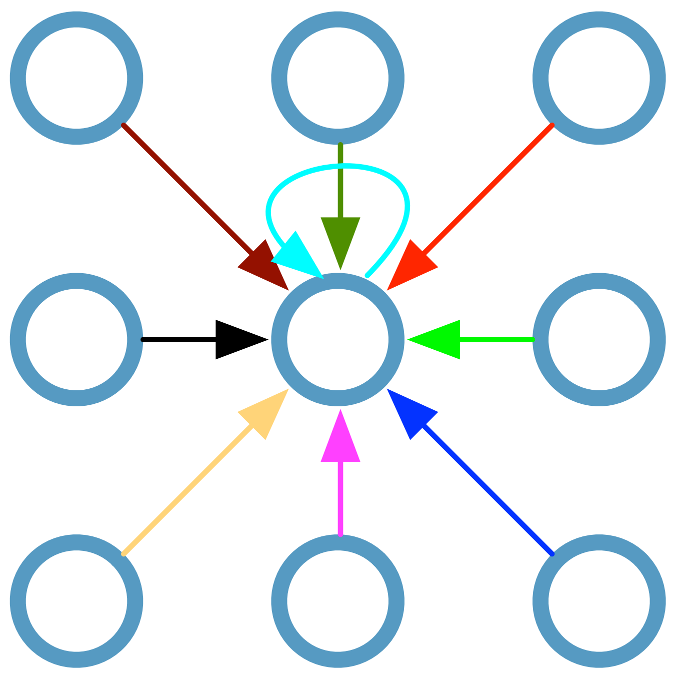


Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:

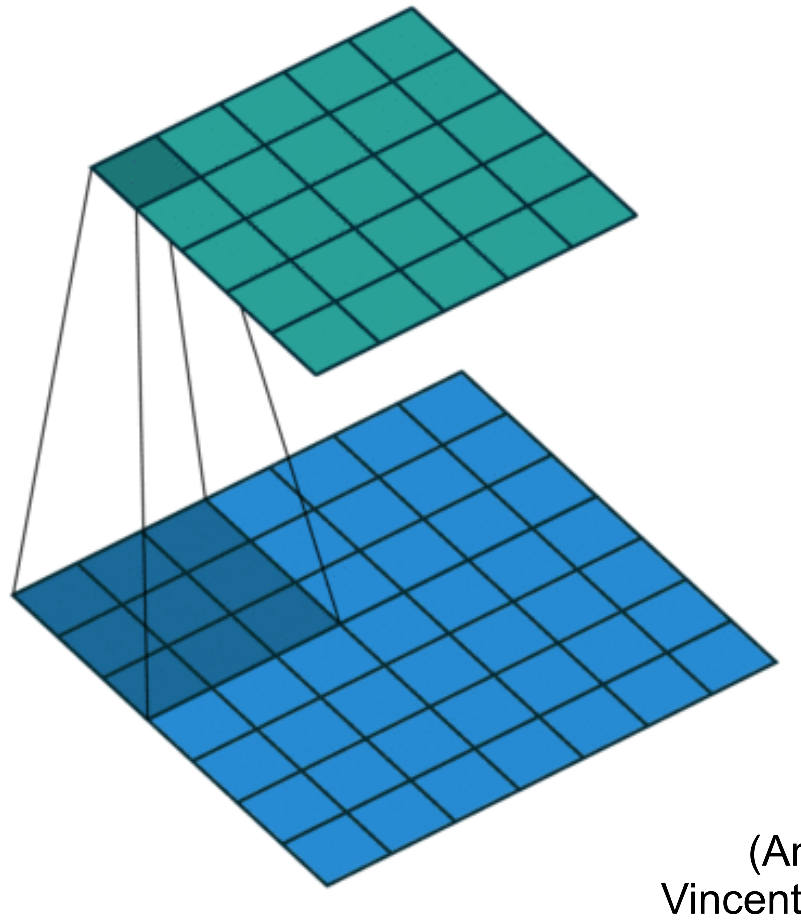


(Animation by Vincent Dumoulin)

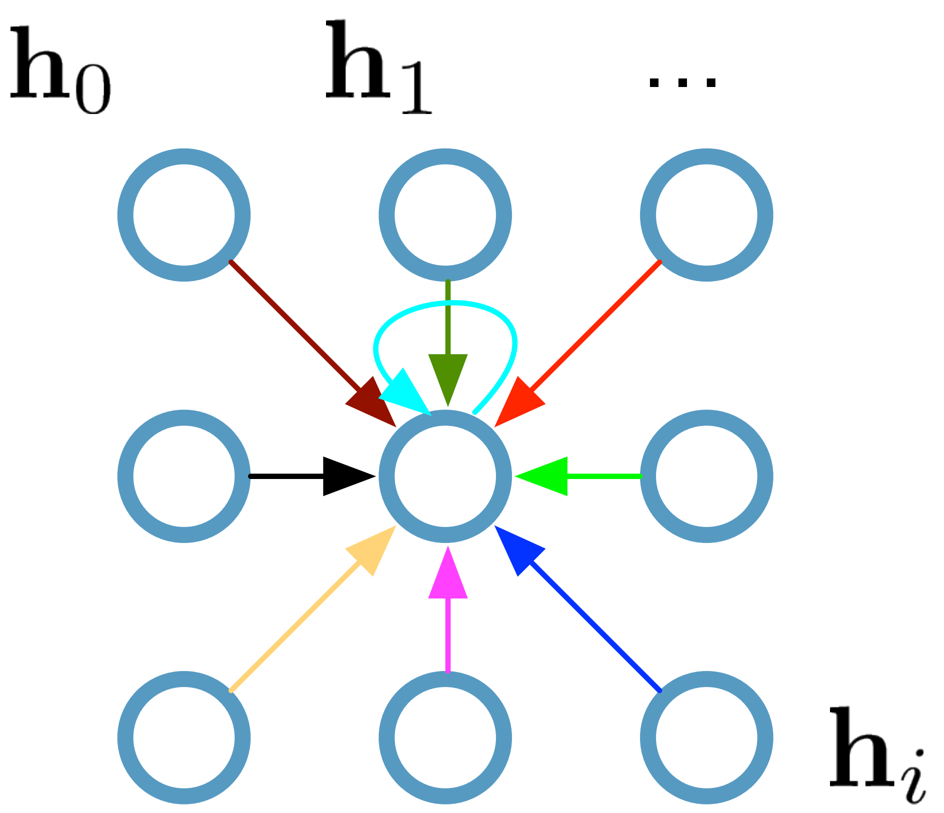


Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:

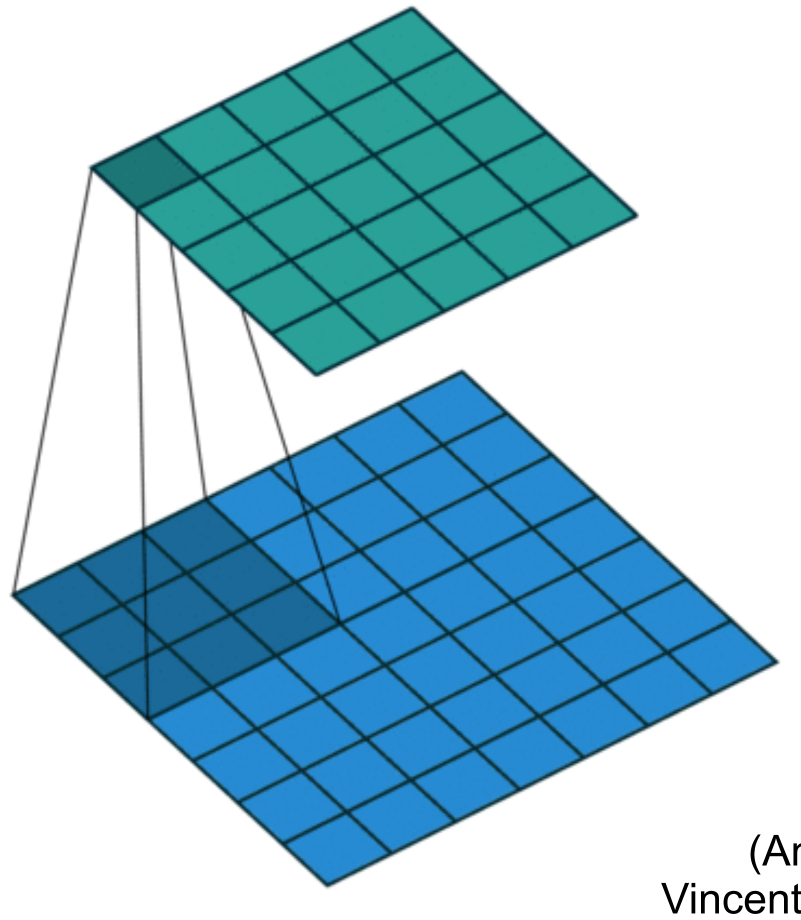


(Animation by Vincent Dumoulin)

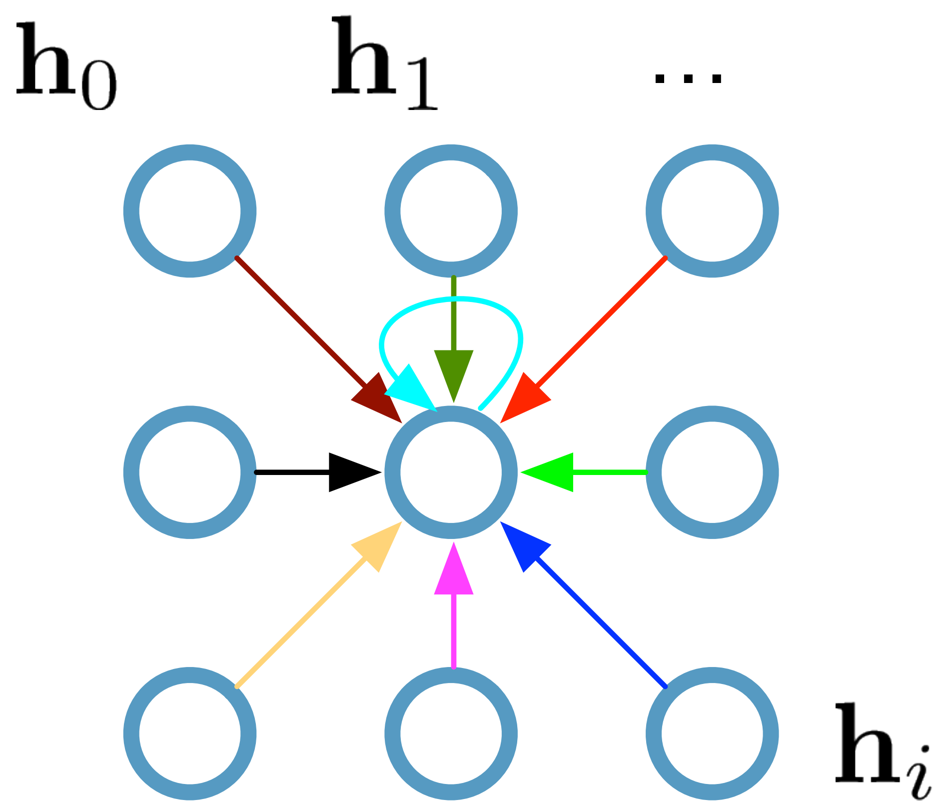


Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:



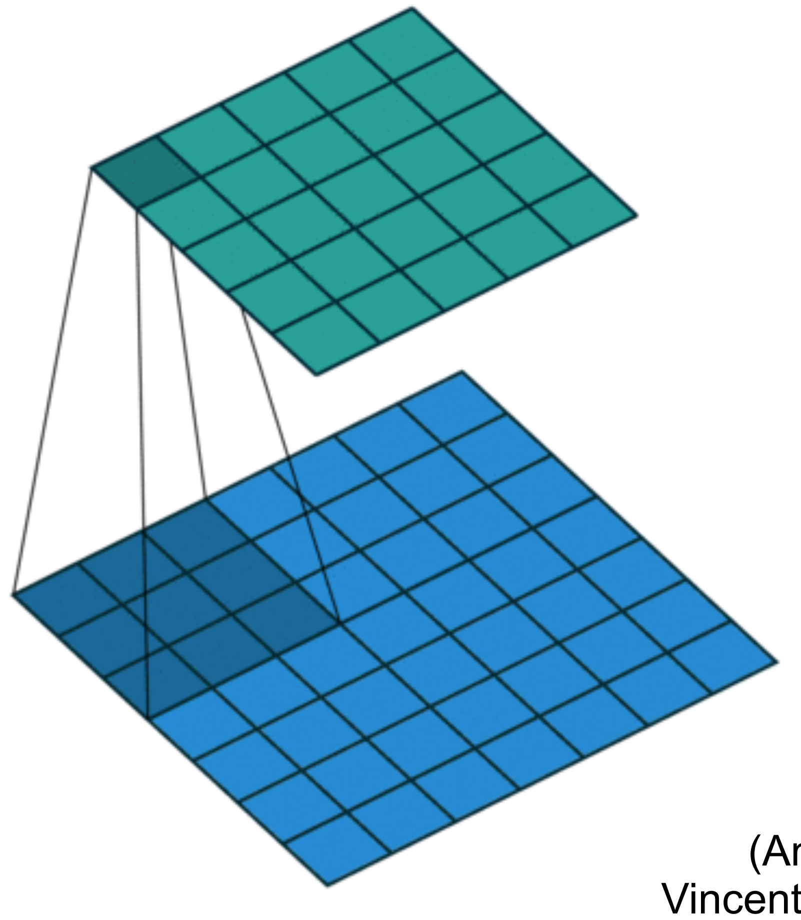
(Animation by Vincent Dumoulin)



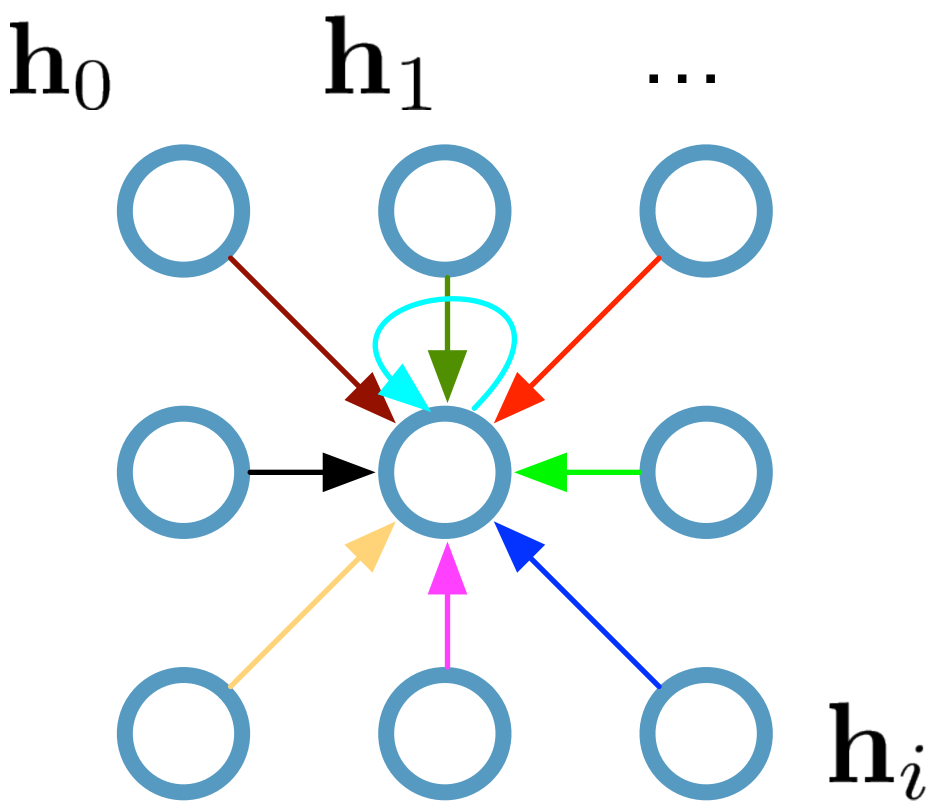
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



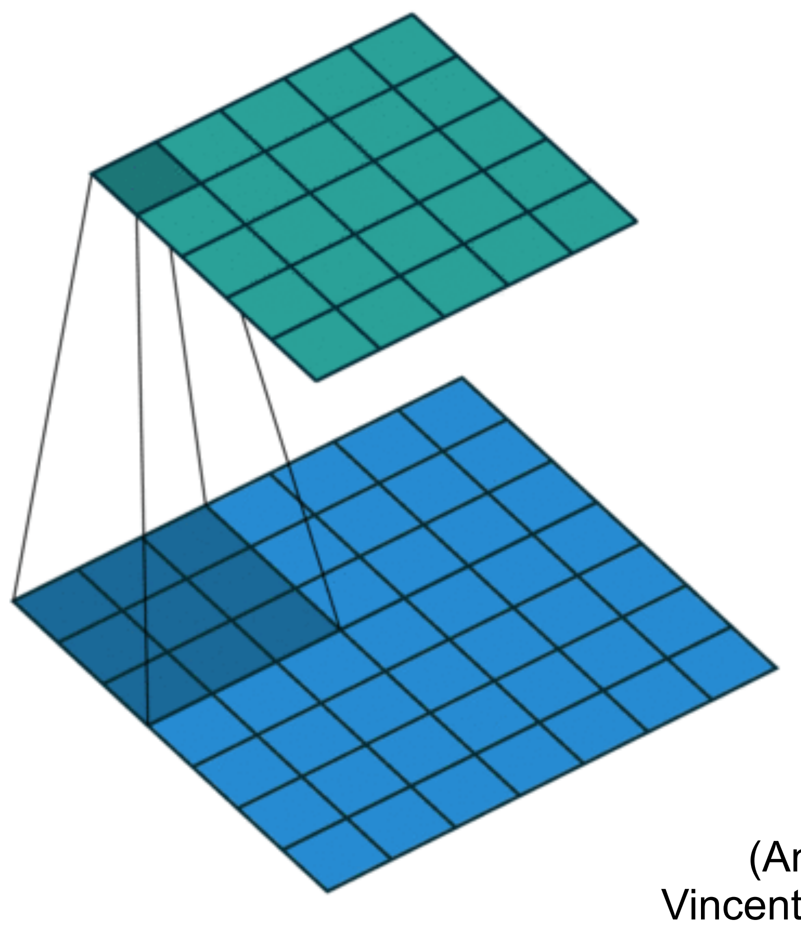
Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

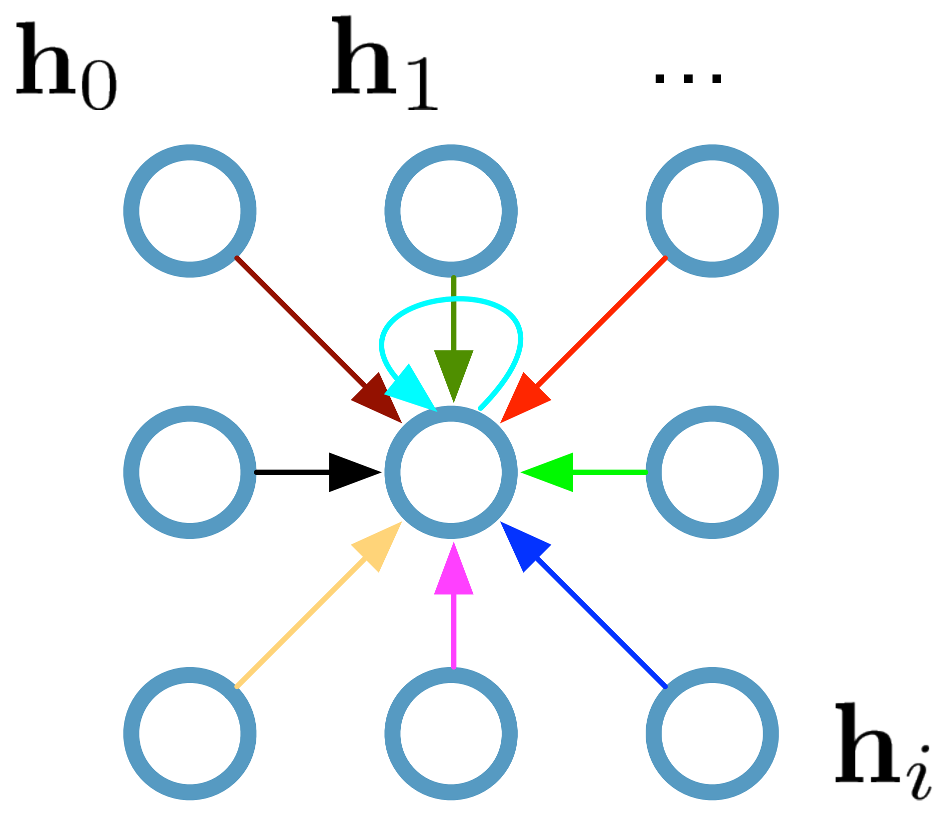
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

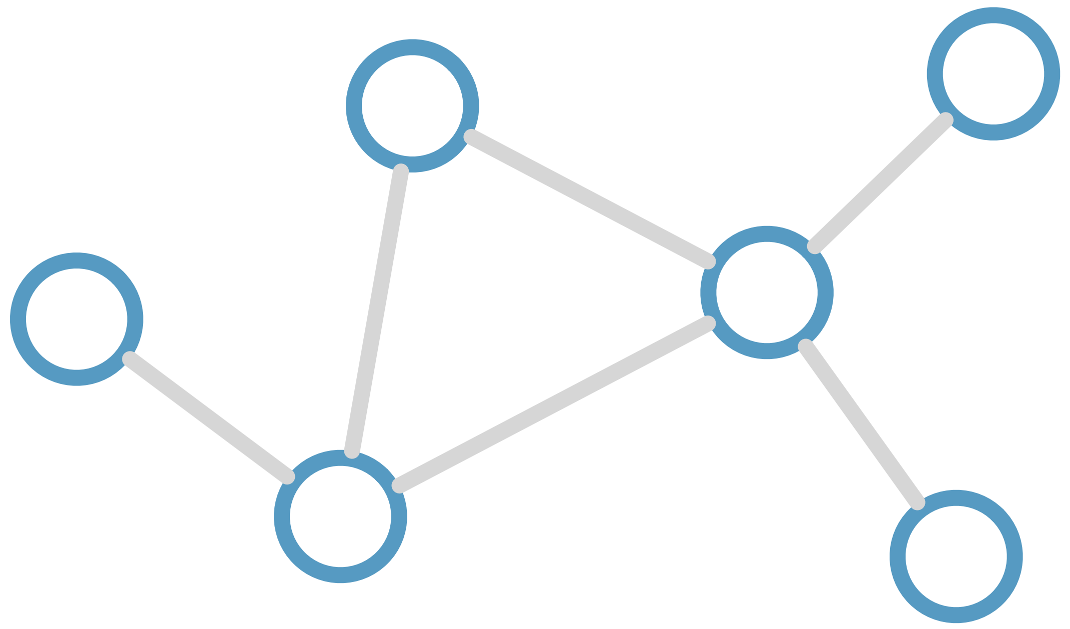
Full update:

$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

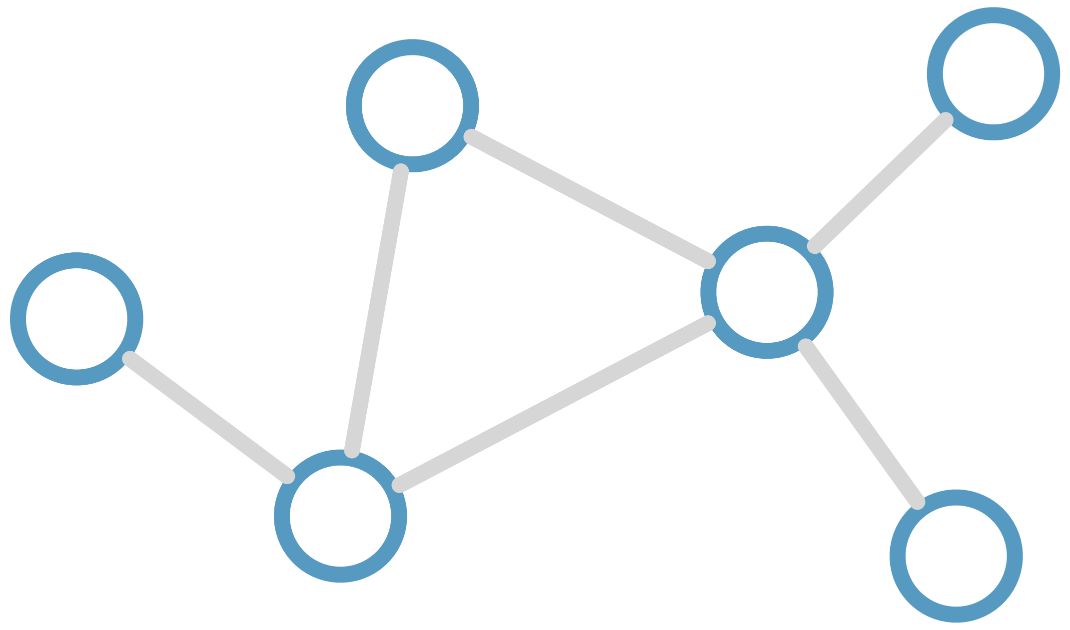
Consider this undirected graph:



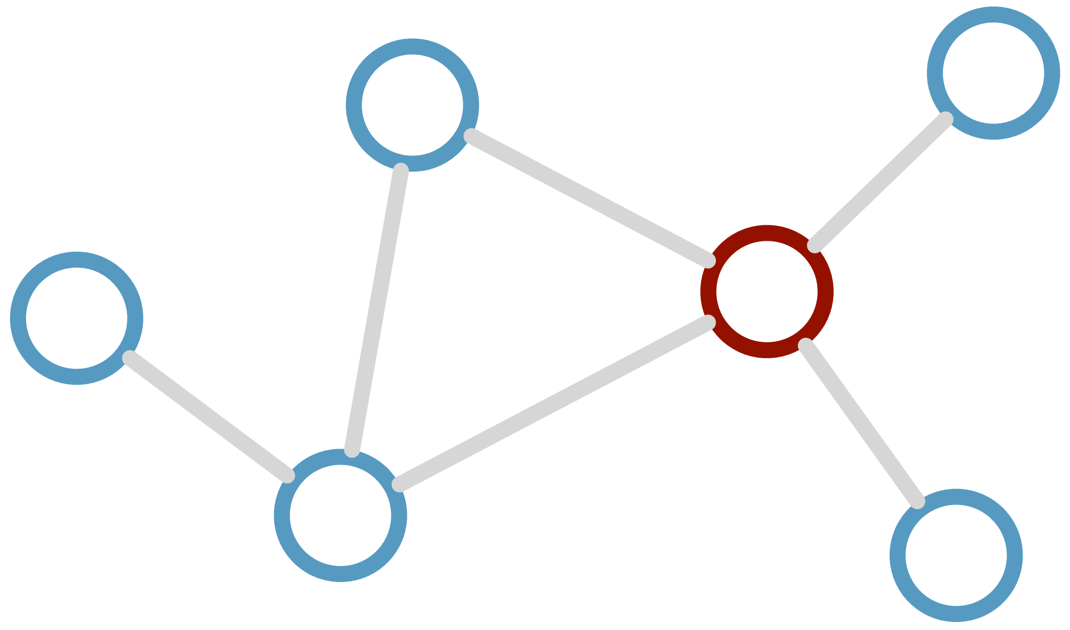
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



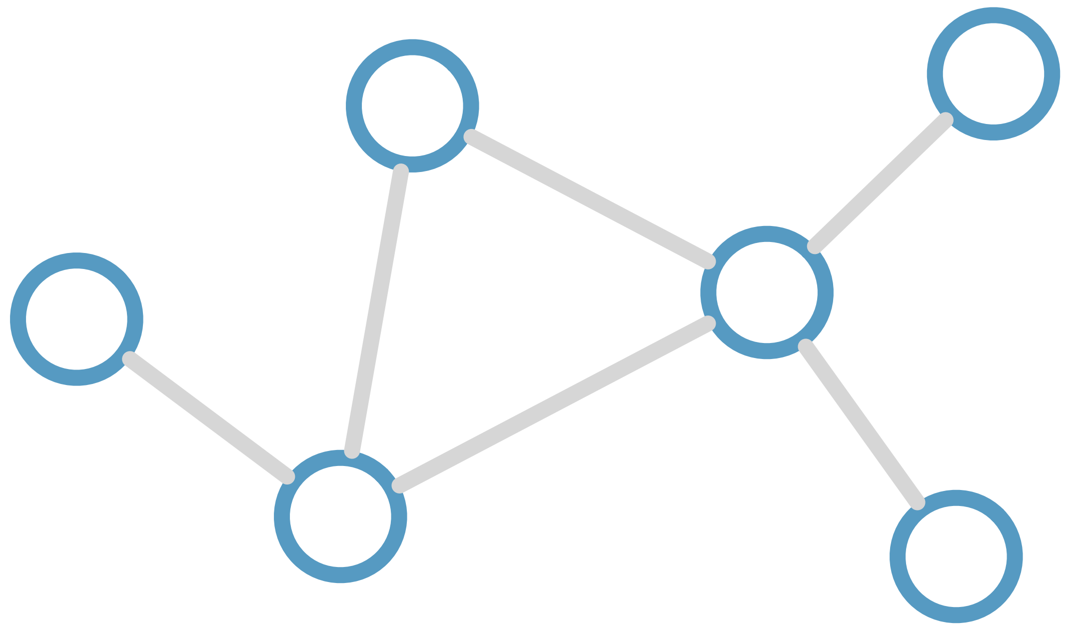
Calculate update for node in red:



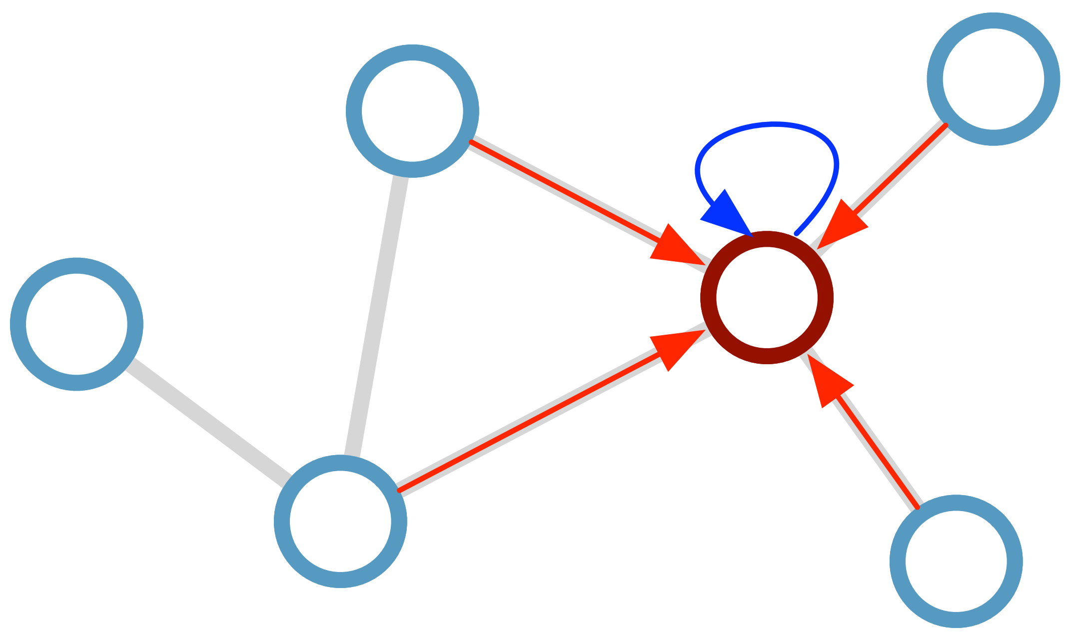
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



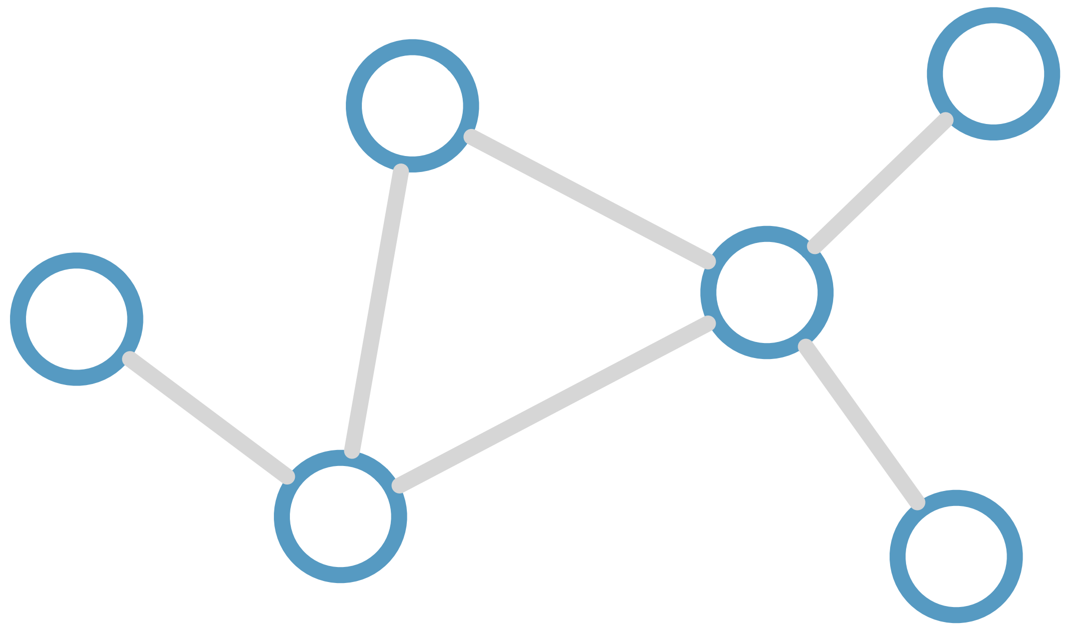
Calculate update for node in red:



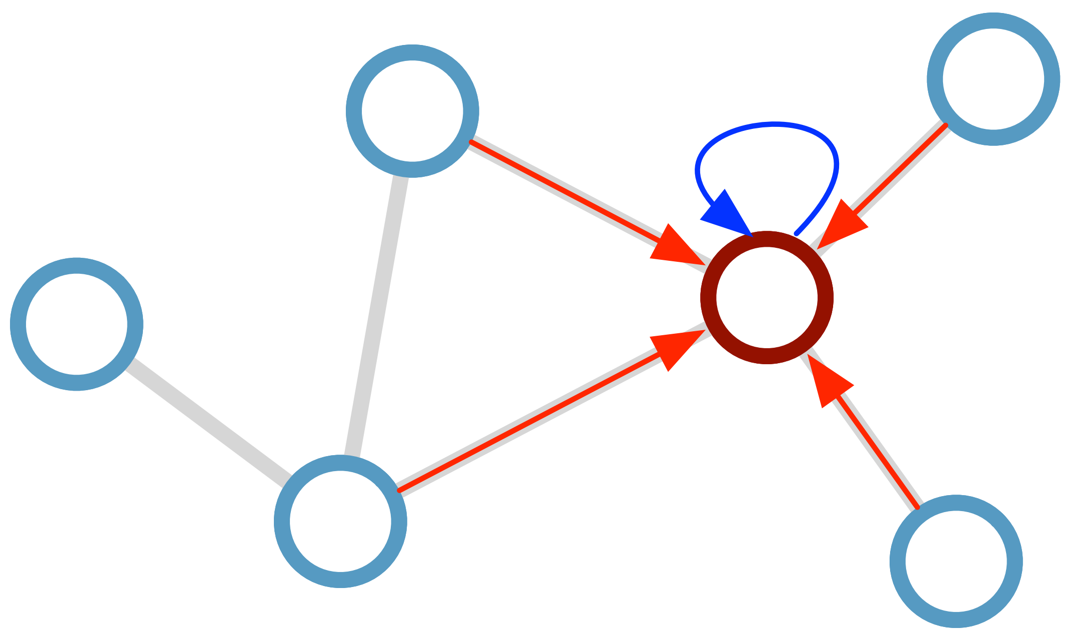
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

$\mathcal{N}_i$: neighbor indices

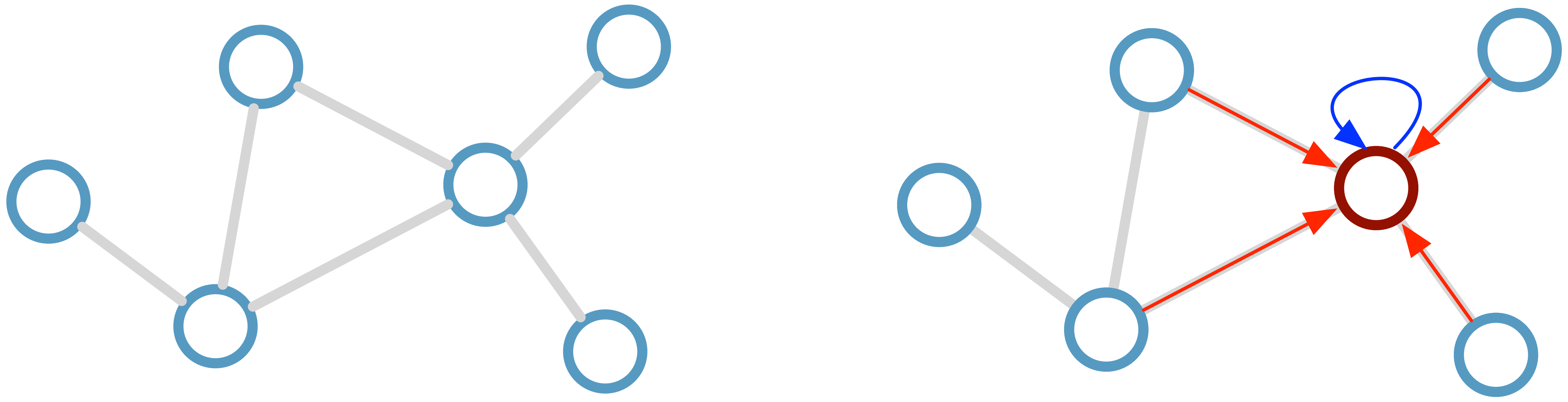
c_{ij} : norm. constant (fixed/trainable)

Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:

Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

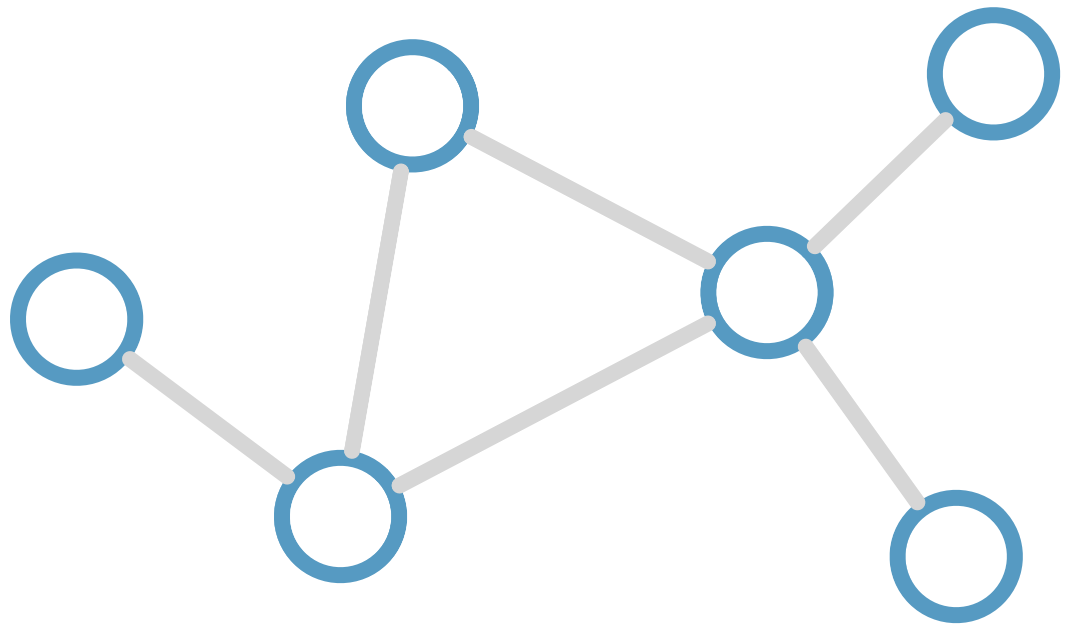
$\mathcal{N}_i$: neighbor indices

c_{ij} : norm. constant (fixed/trainable)

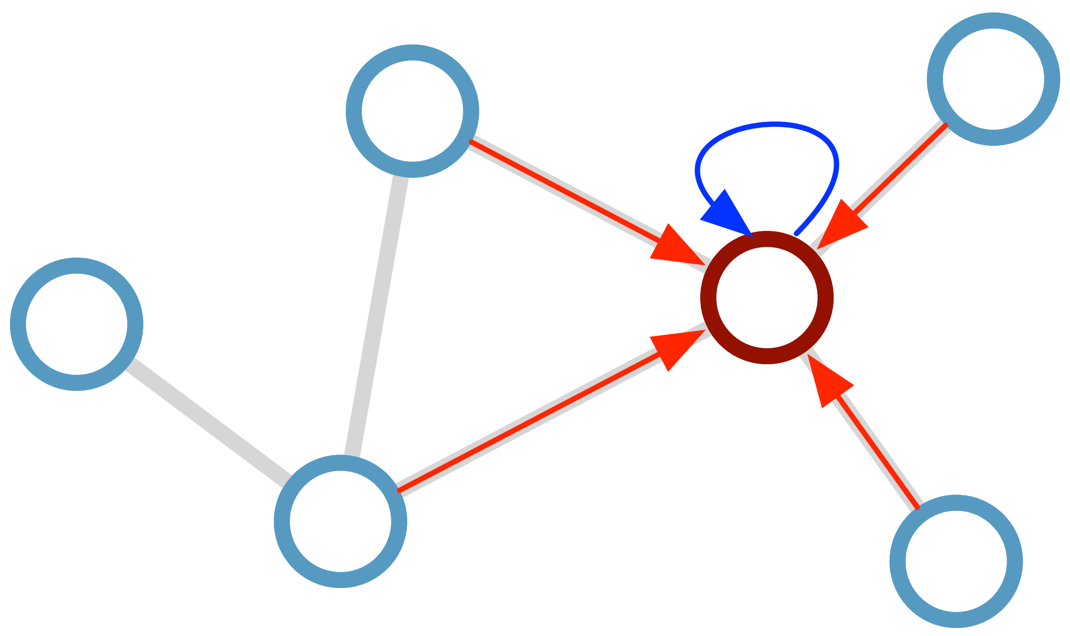
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity O(E)
- Applicable both in transductive and inductive settings

Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

$\mathcal{N}_i$: neighbor indices

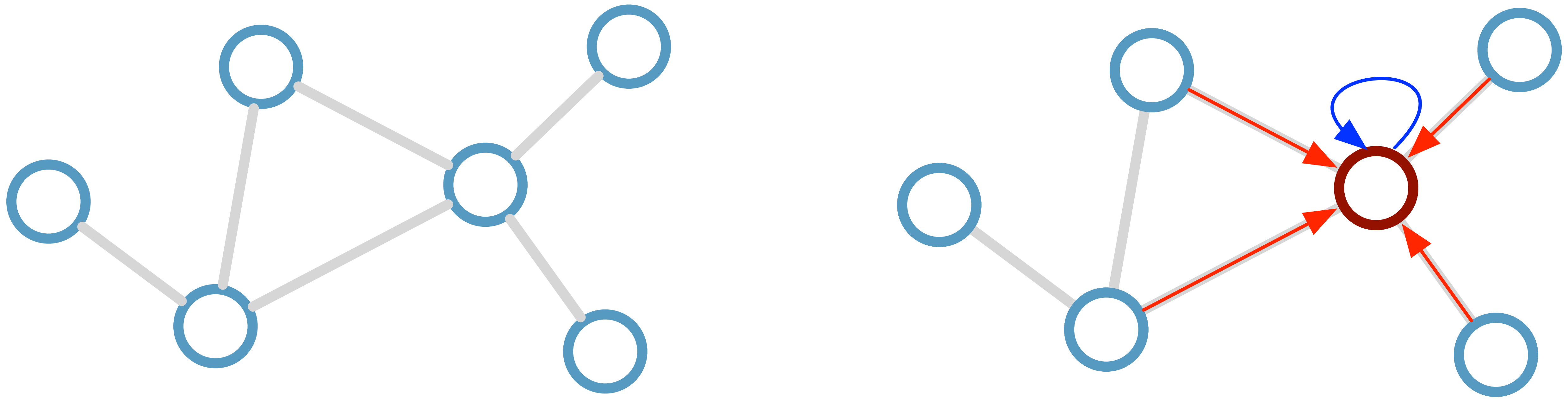
c_{ij} : norm. constant (fixed/trainable)

Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:

Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity O(E)
- Applicable both in transductive and inductive settings

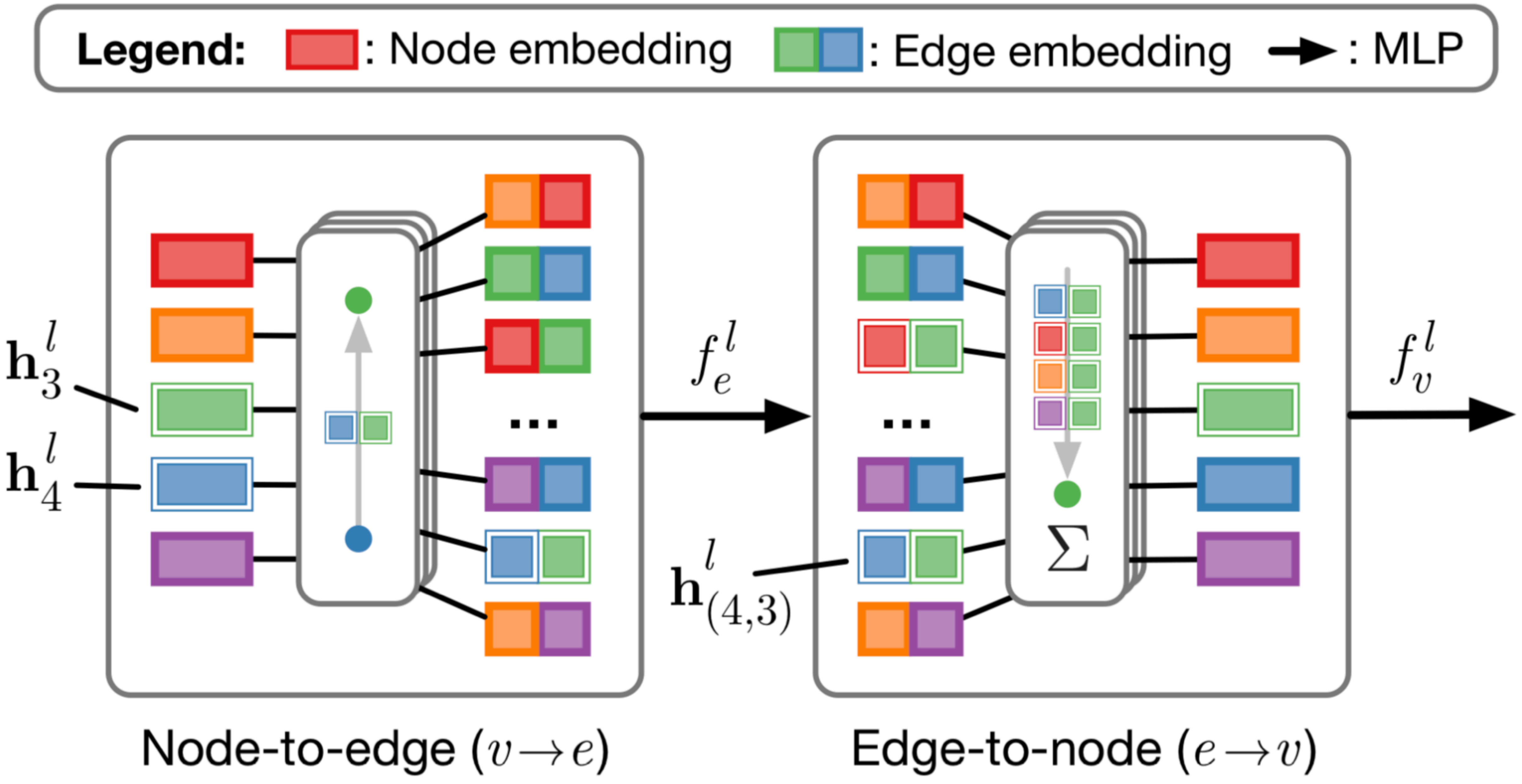
Limitations:

- Requires gating mechanism / residual connections for depth
- Only indirect support for edge features

$\mathcal{N}_i$: neighbor indices c_{ij} : norm. constant (fixed/trainable)

GNNs with edge embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



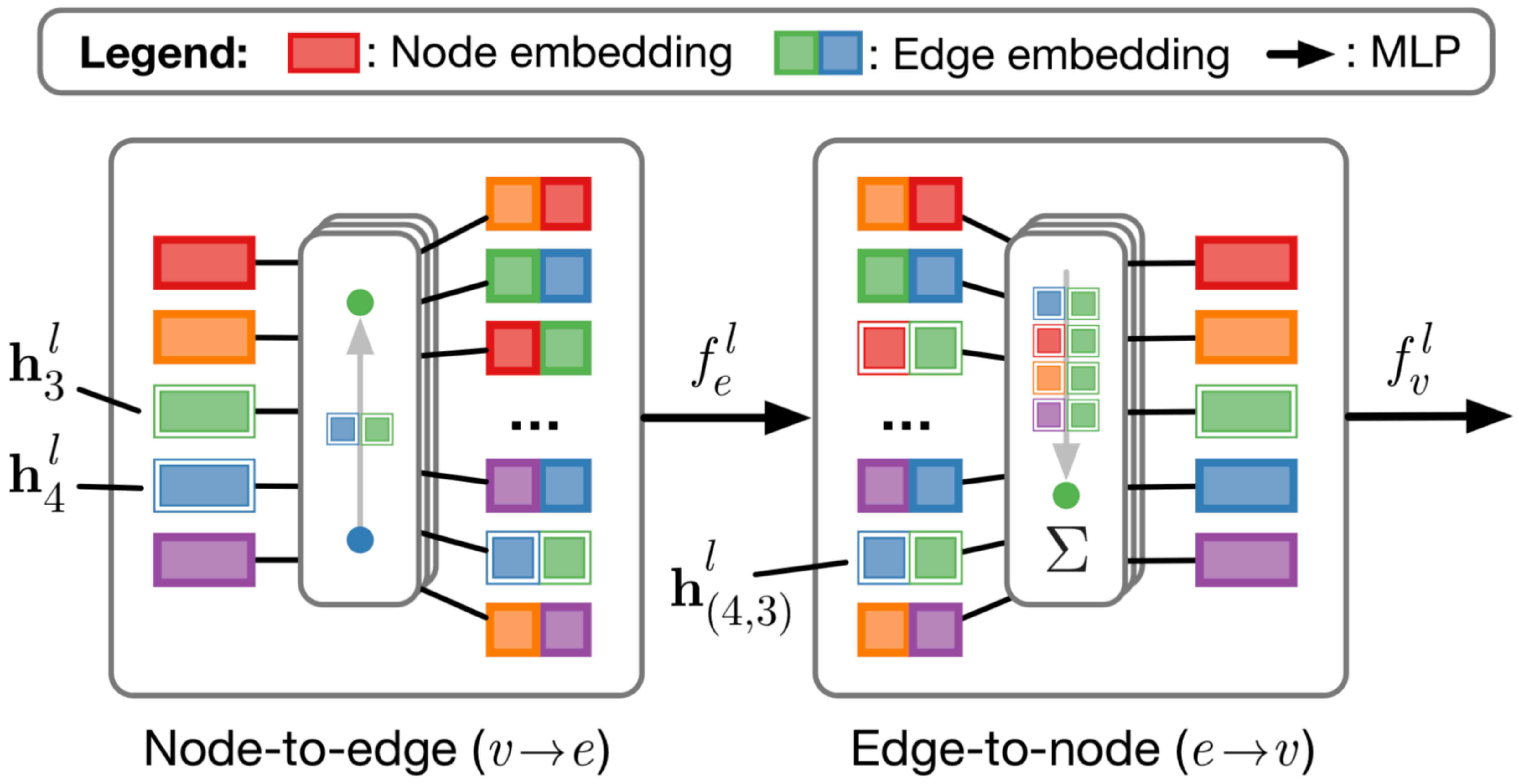
Formally:

$$v \to e: \quad \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$$

$$e \to v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

GNNs with edge embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



Pros:

- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

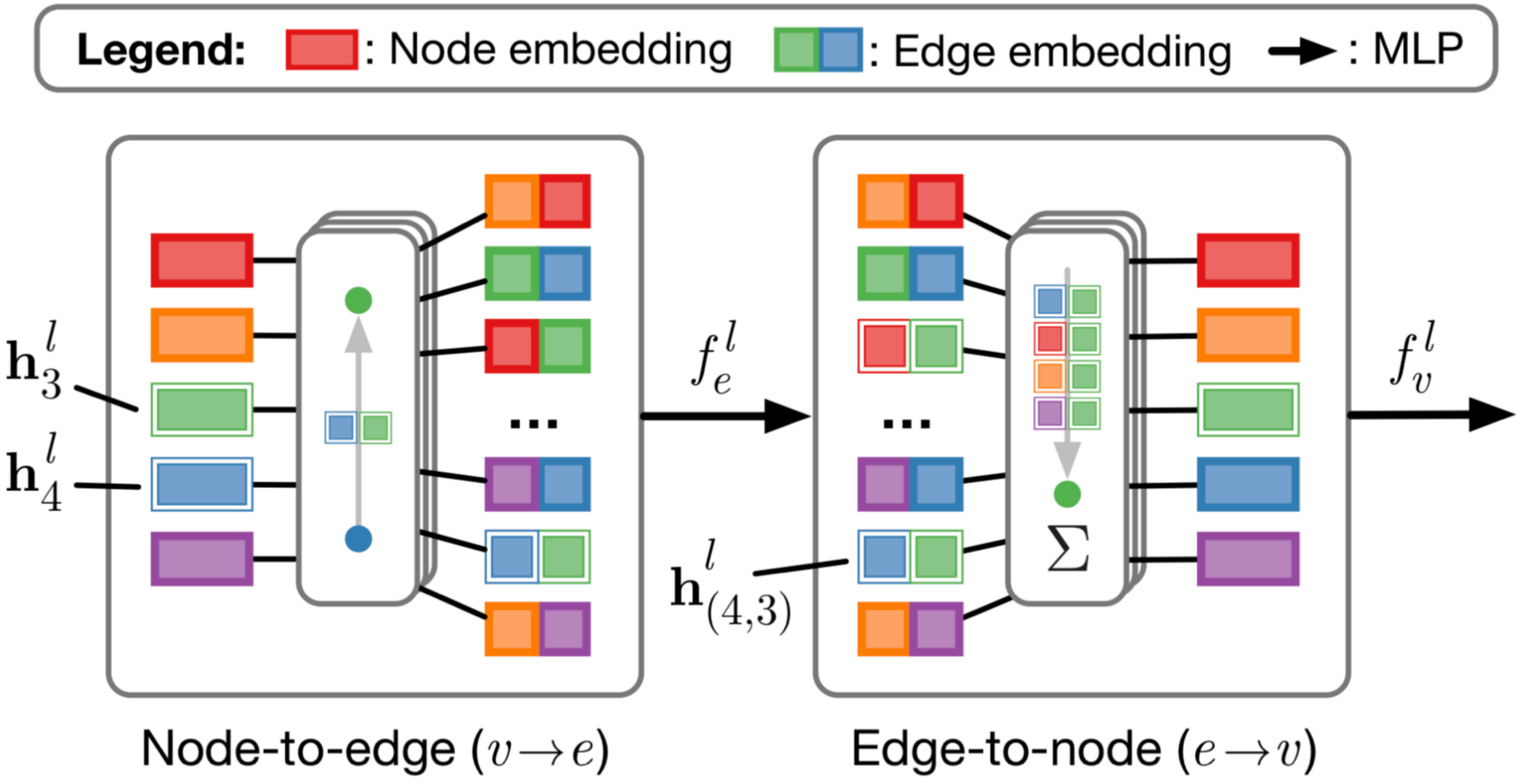
Formally:

$$v\rightarrow e: \quad \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$$

$$e\rightarrow v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i\in\mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

GNNs with edge embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



Formally:

$$v \rightarrow e: \quad \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$$

$$e \rightarrow v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

Pros:

- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

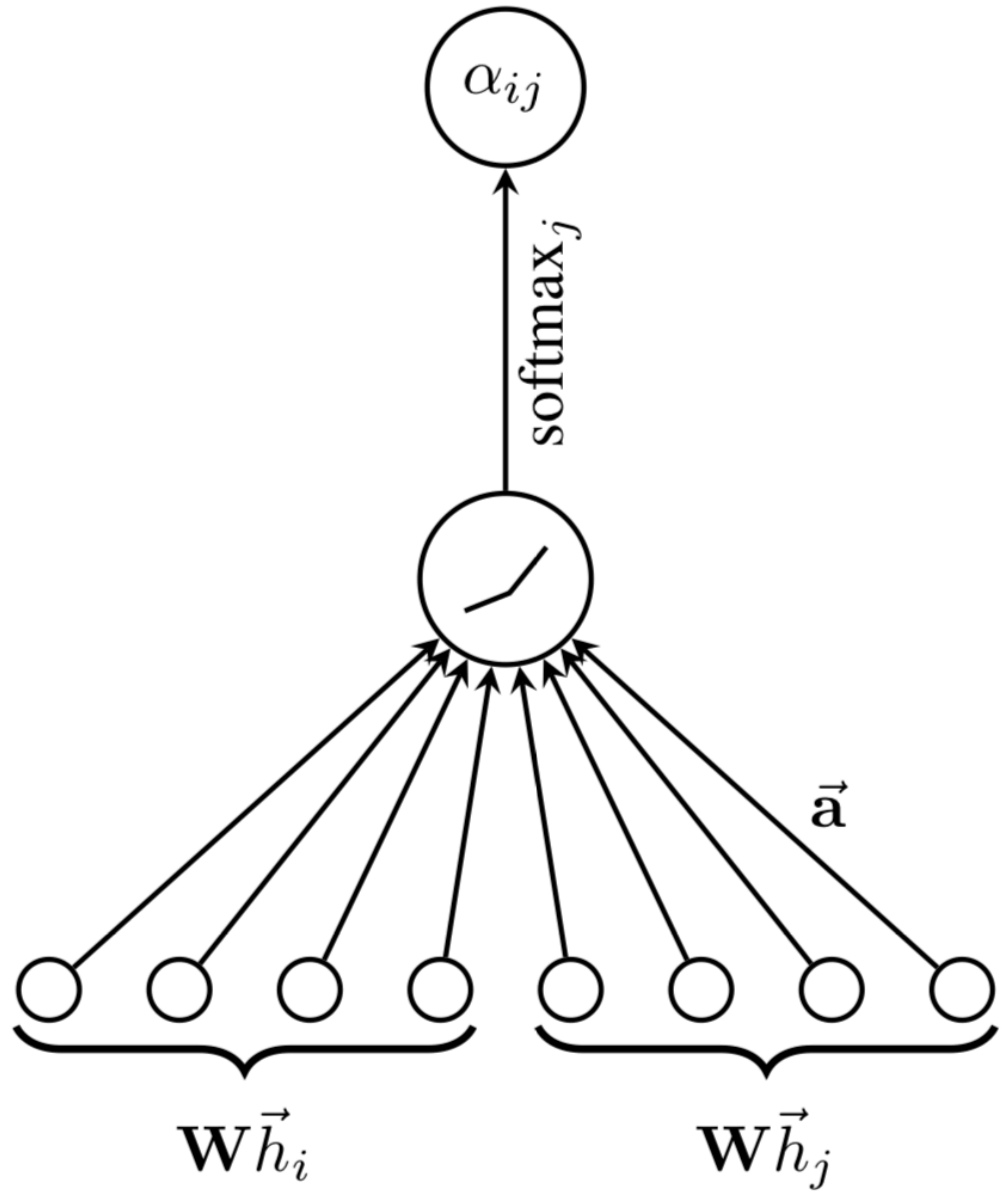
Cons:

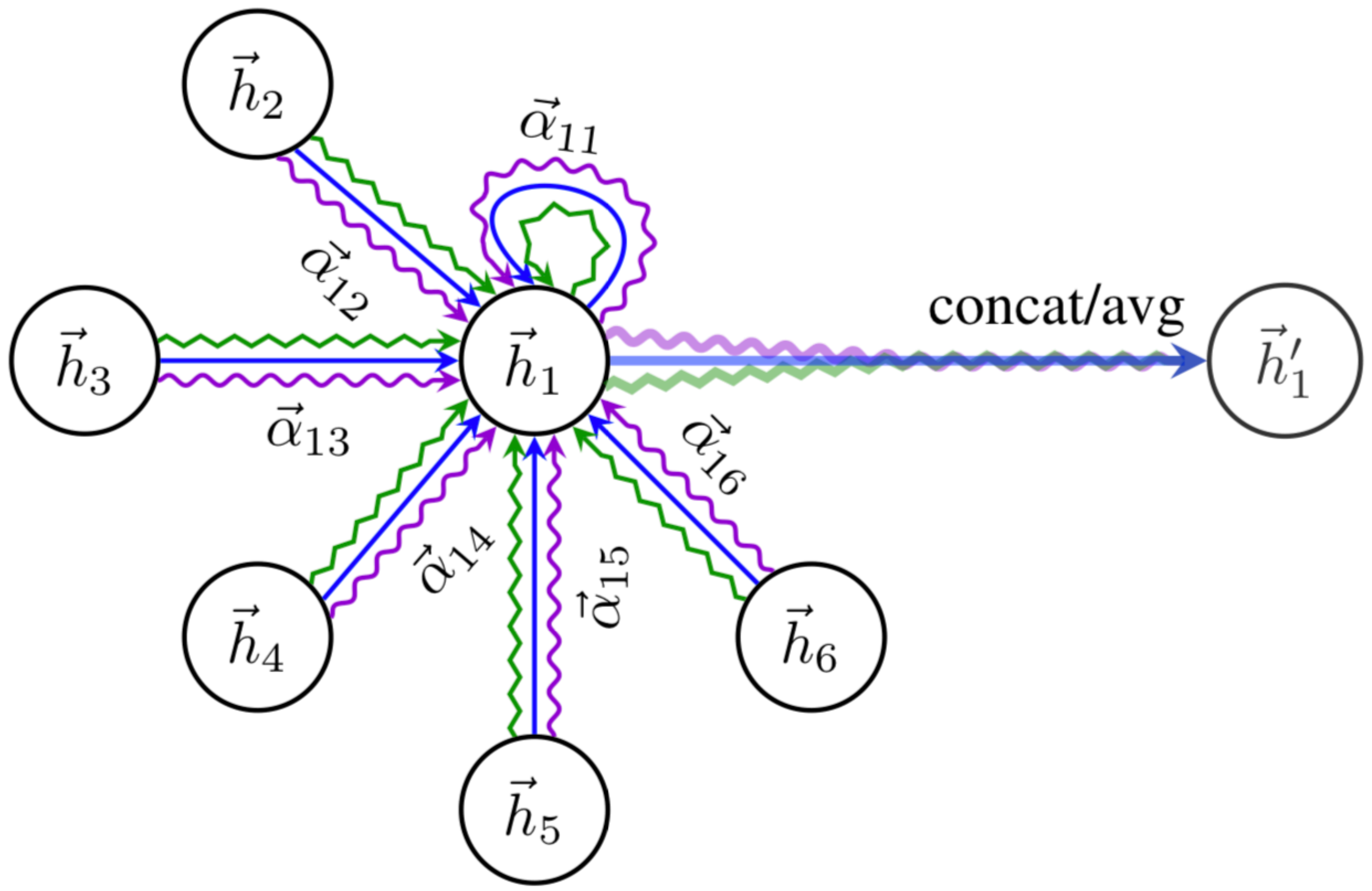
- Need to store intermediate edge-based activations
- Difficult to implement with subsampling

➡ In practice limited to small graphs

Graph neural networks with attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



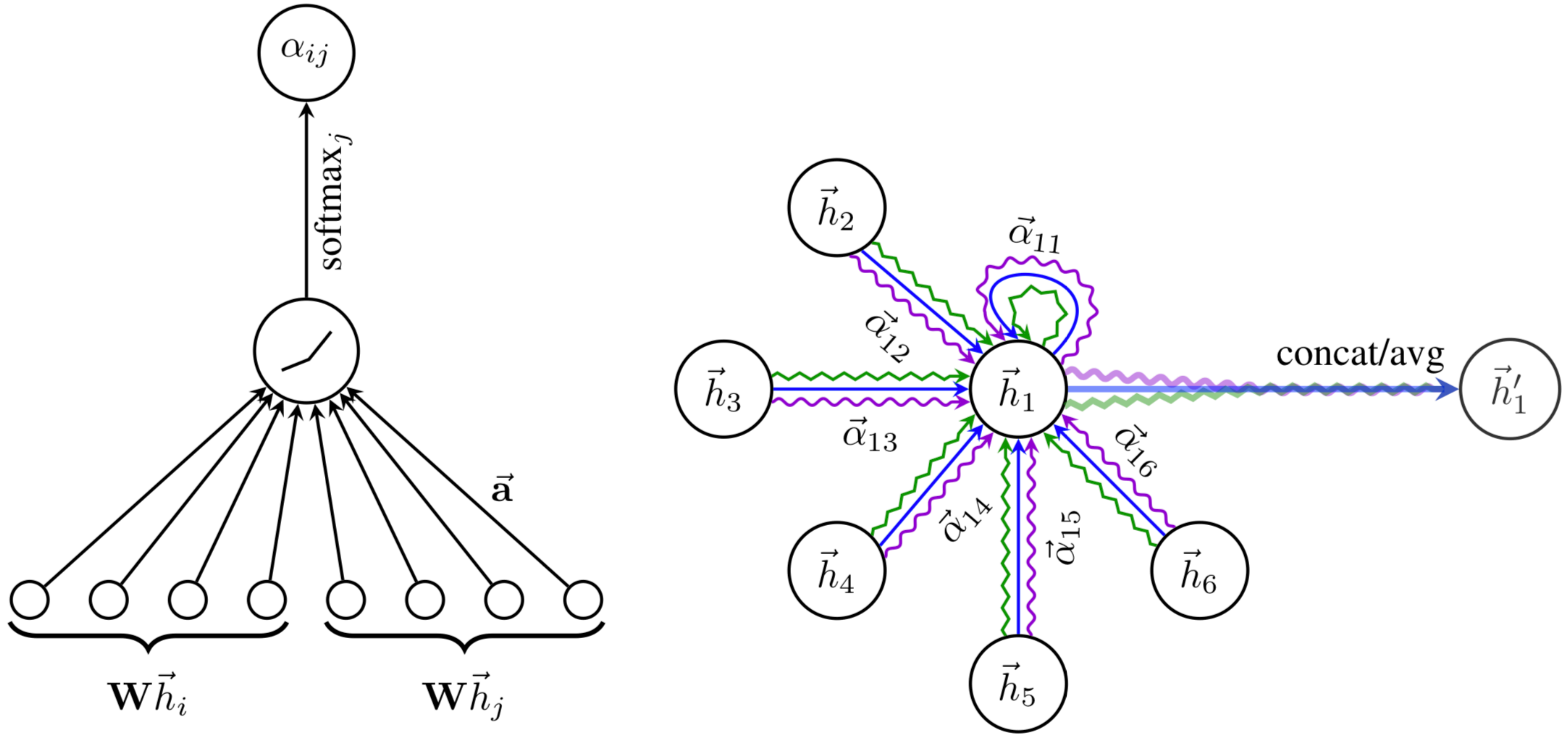


[Figure from Veličković et al. (ICLR 2018)]

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

Graph neural networks with attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)

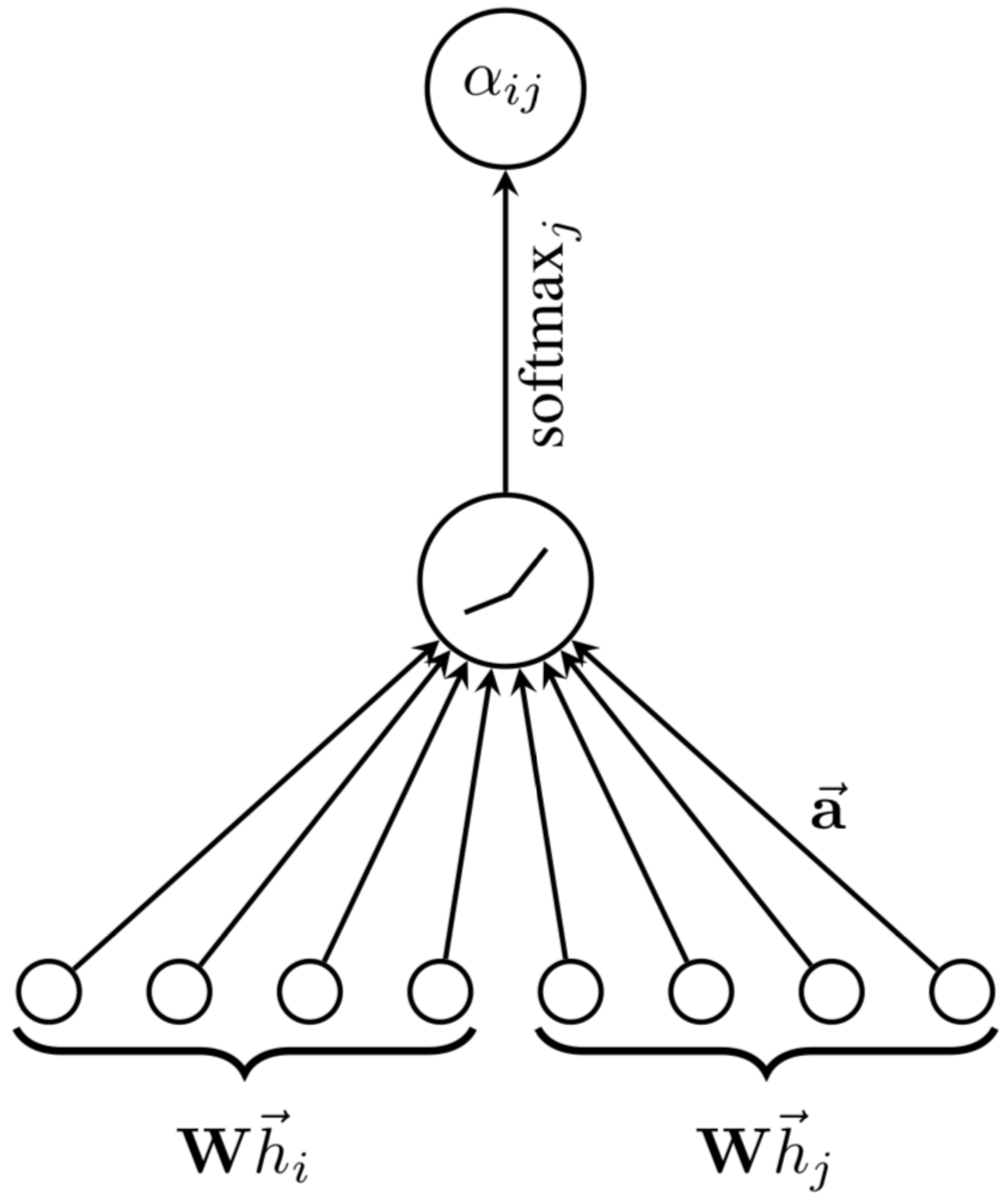


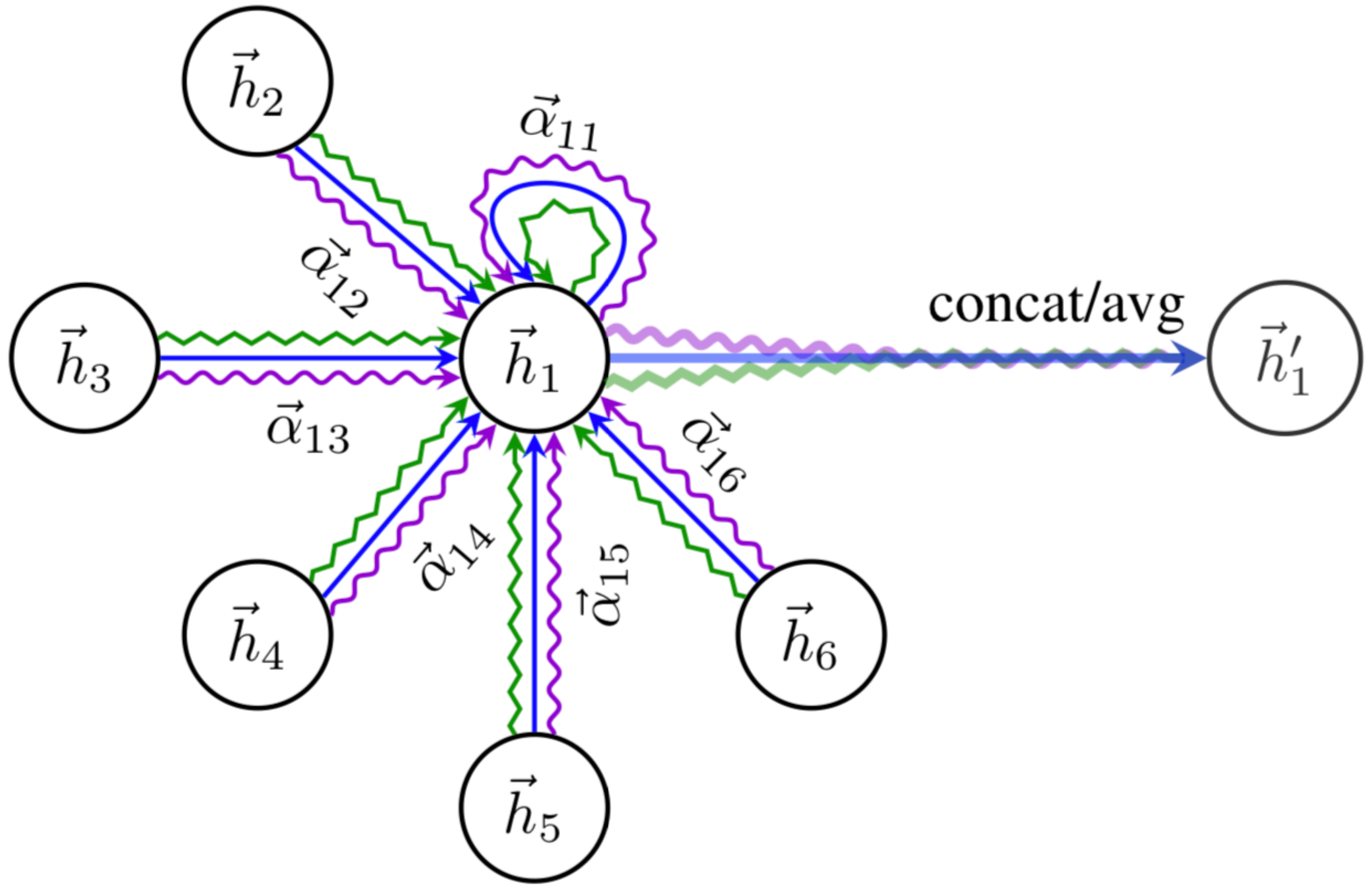
[Figure from Veličković et al. (ICLR 2018)]

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right) \qquad \alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_k]\right)\right)}$$

Graph neural networks with attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)





[Figure from Veličković et al. (ICLR 2018)]

Pros:

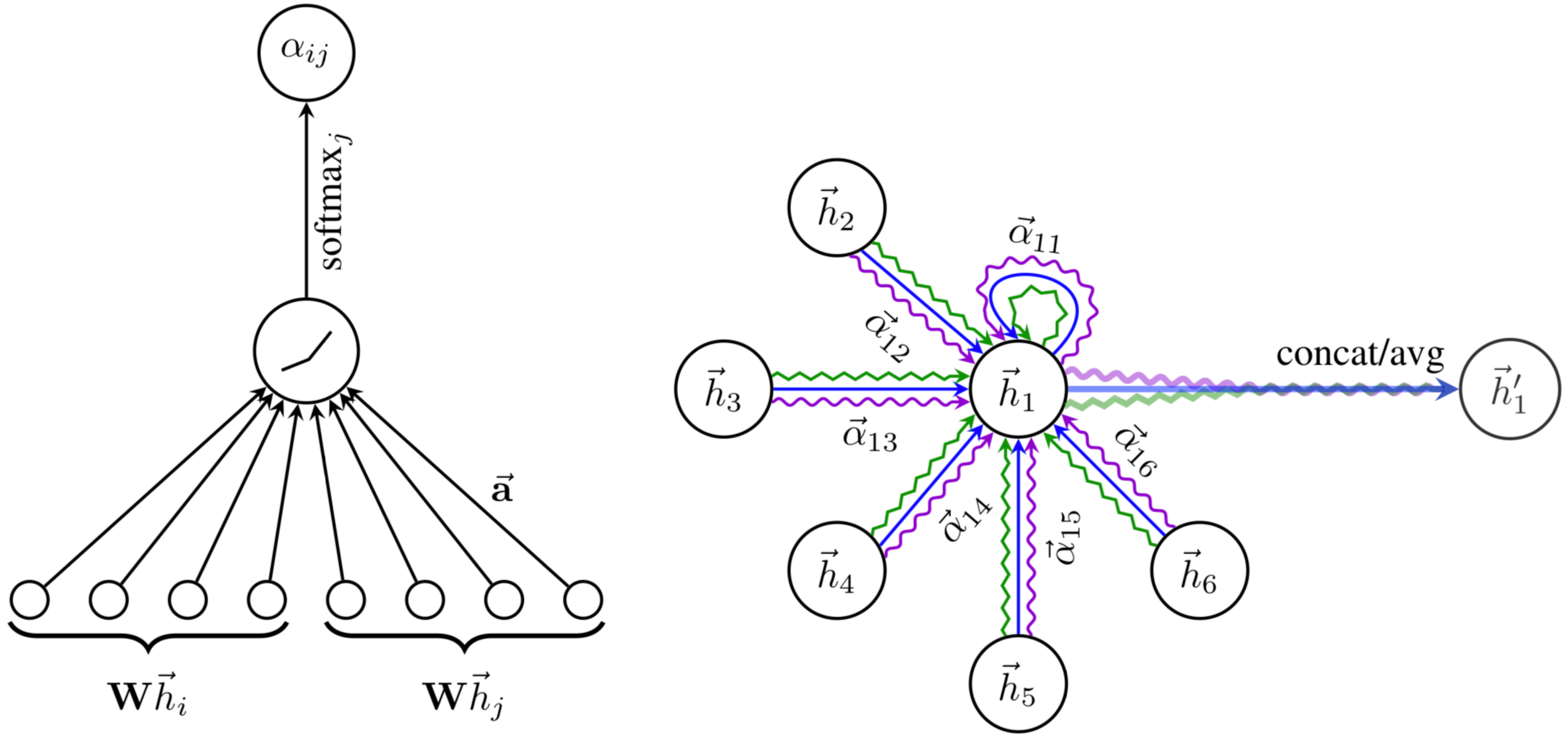
- No need to store intermediate edge-based activation vectors (when using dot-product attn.)
- Slower than GCNs but faster than GNNs with edge embeddings

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Graph neural networks with attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



[Figure from Veličković et al. (ICLR 2018)]

Pros:

- No need to store intermediate edge-based activation vectors (when using dot-product attn.)
- Slower than GCNs but faster than GNNs with edge embeddings

Cons:

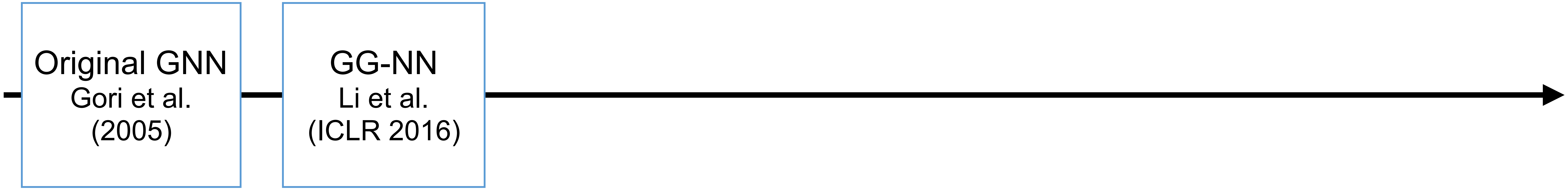
- (Most likely) less expressive than GNNs with edge embeddings
- Can be more difficult to optimize

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

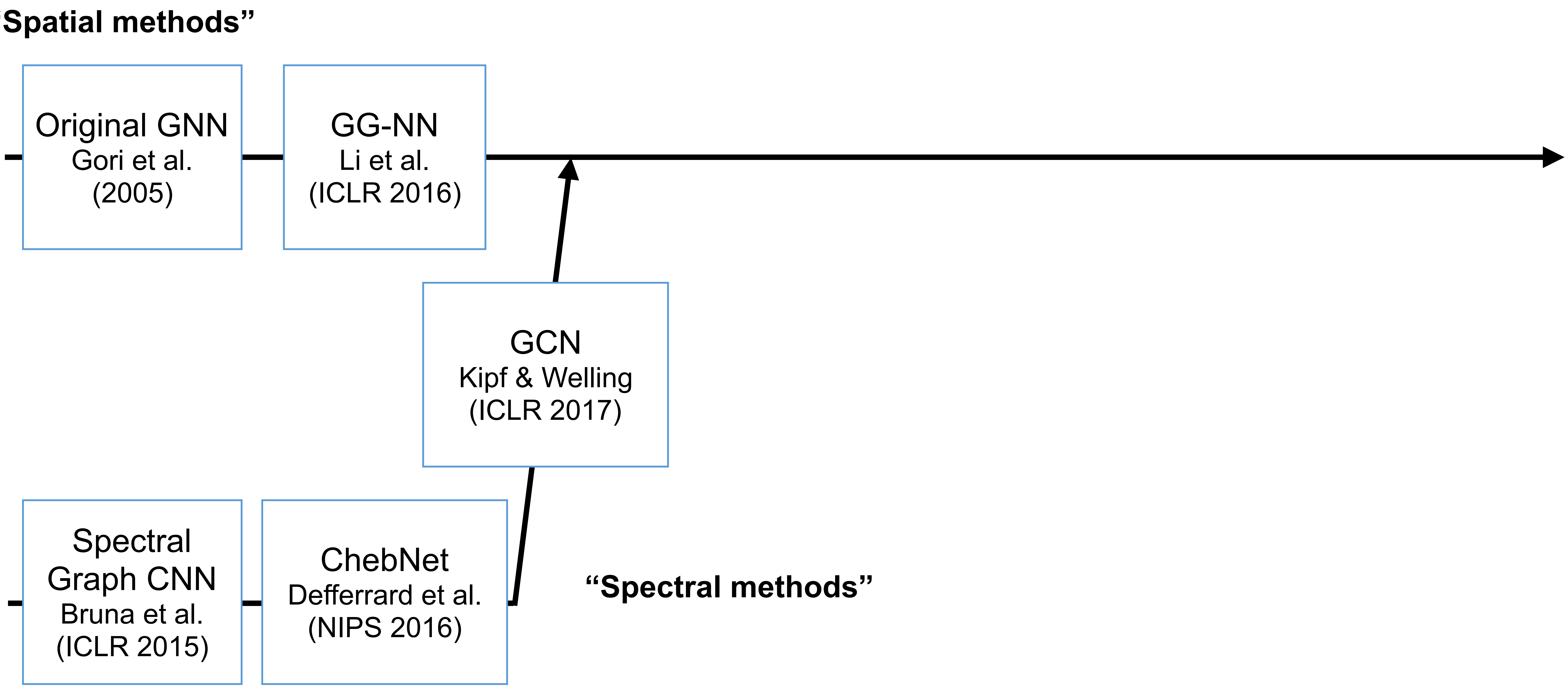
A brief history of graph neural nets

“Spatial methods”



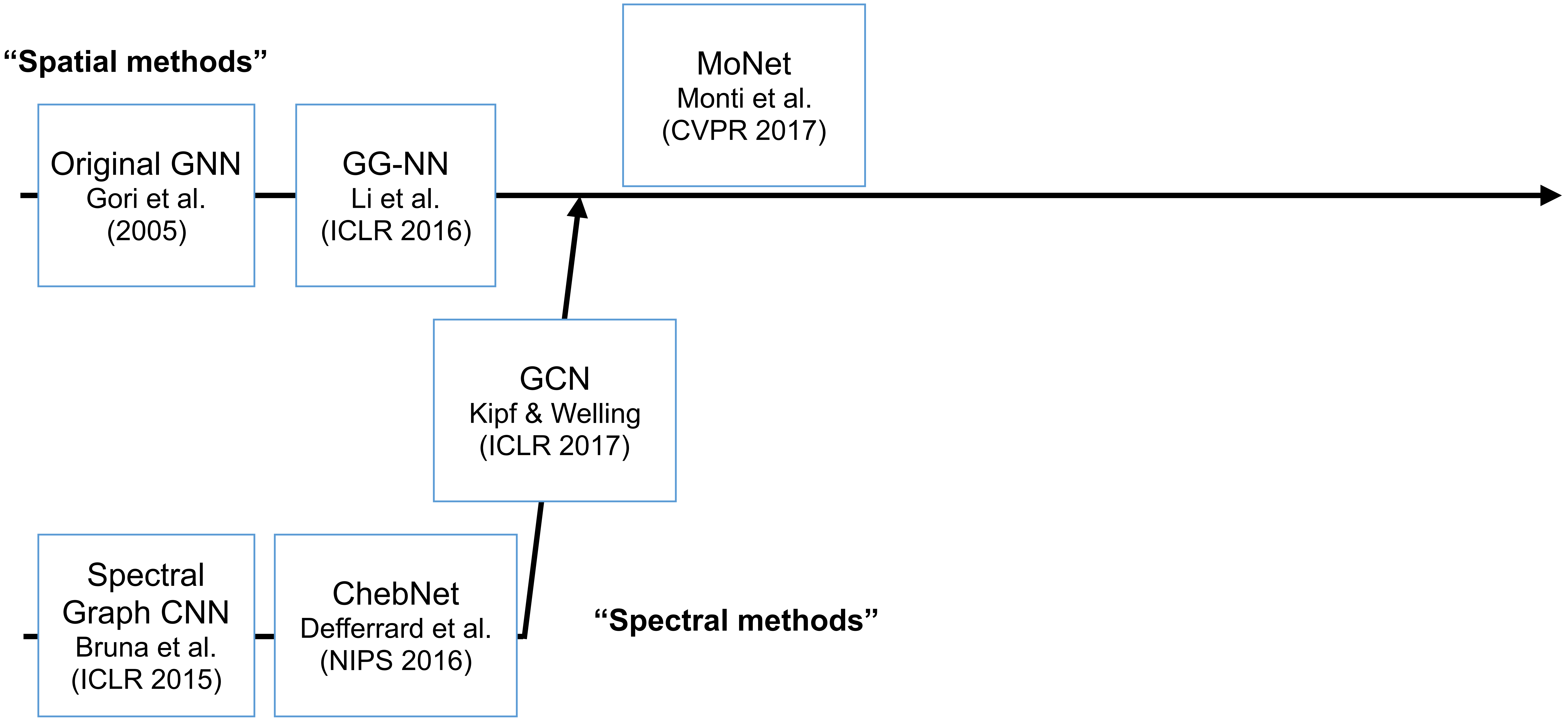
(slide inspired by Alexander Gaunt’s talk on GNNs)

A brief history of graph neural nets



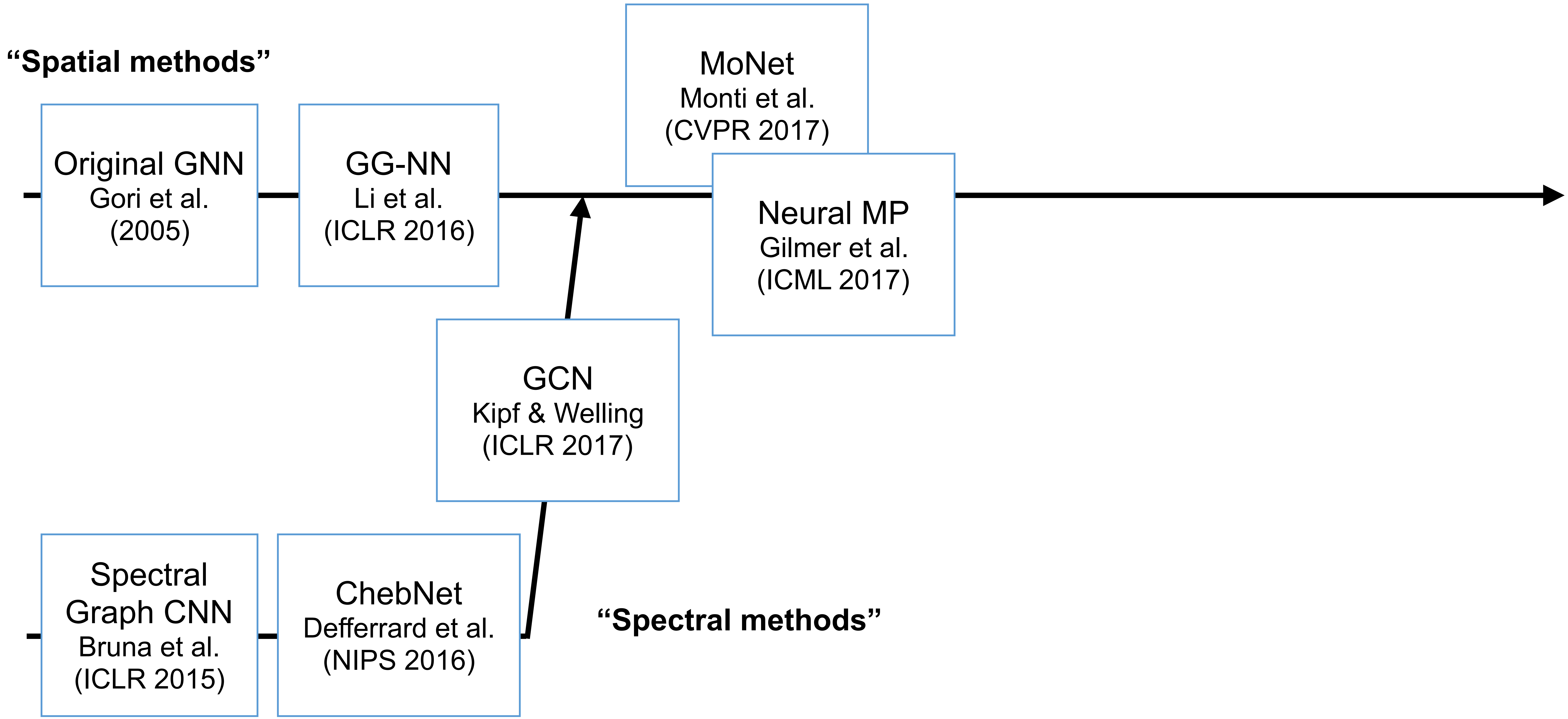
(slide inspired by Alexander Gaunt’s talk on GNNs)

A brief history of graph neural nets



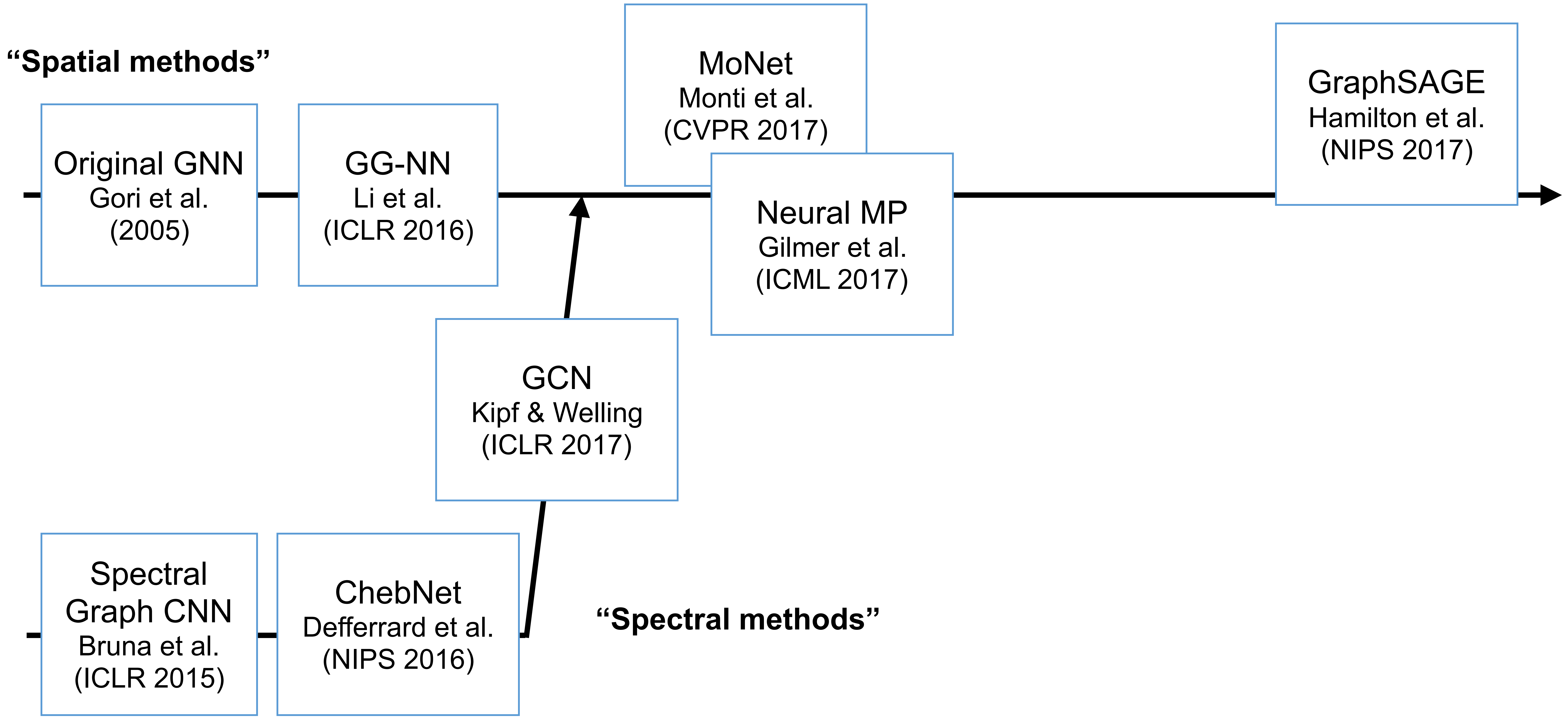
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



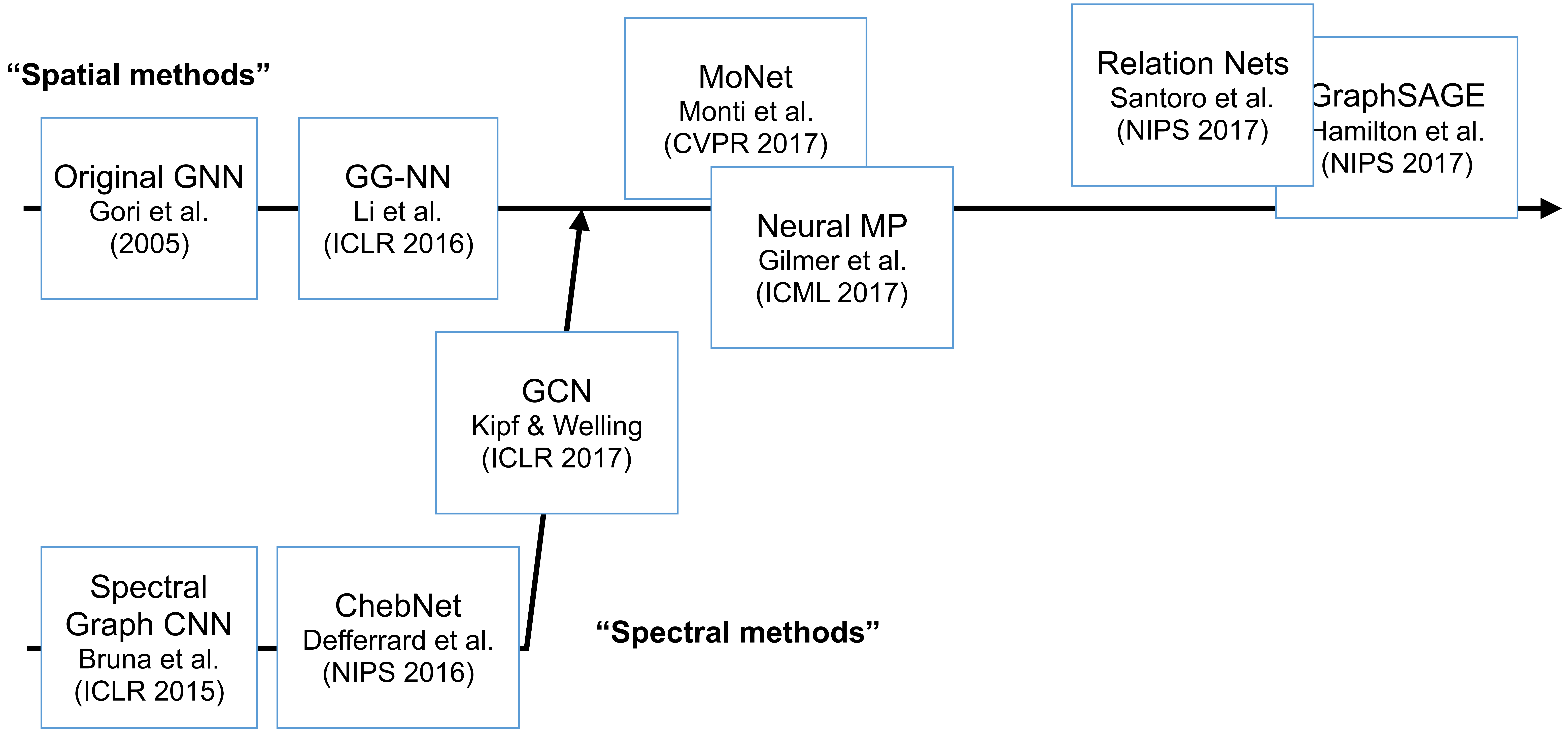
(slide inspired by Alexander Gaunt’s talk on GNNs)

A brief history of graph neural nets



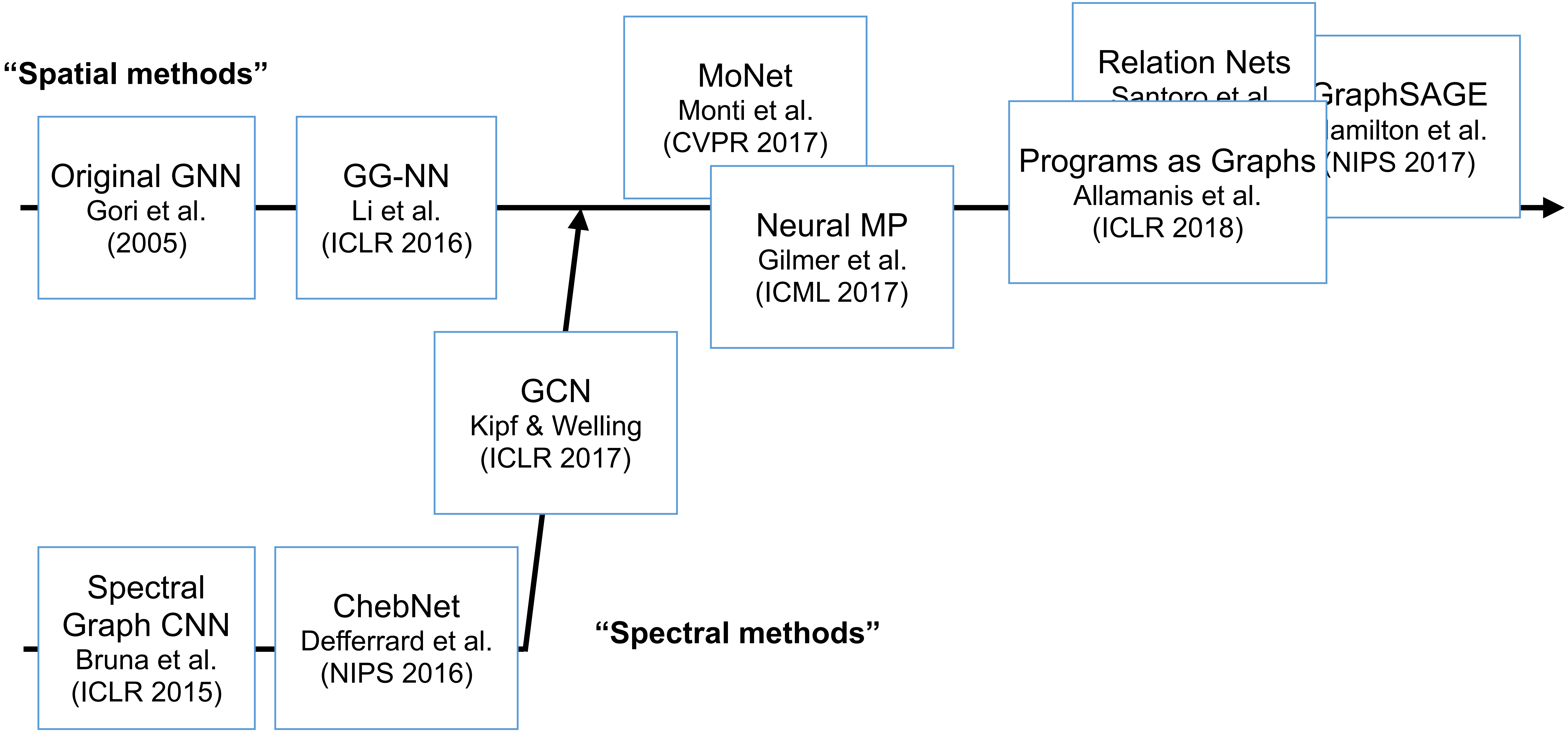
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



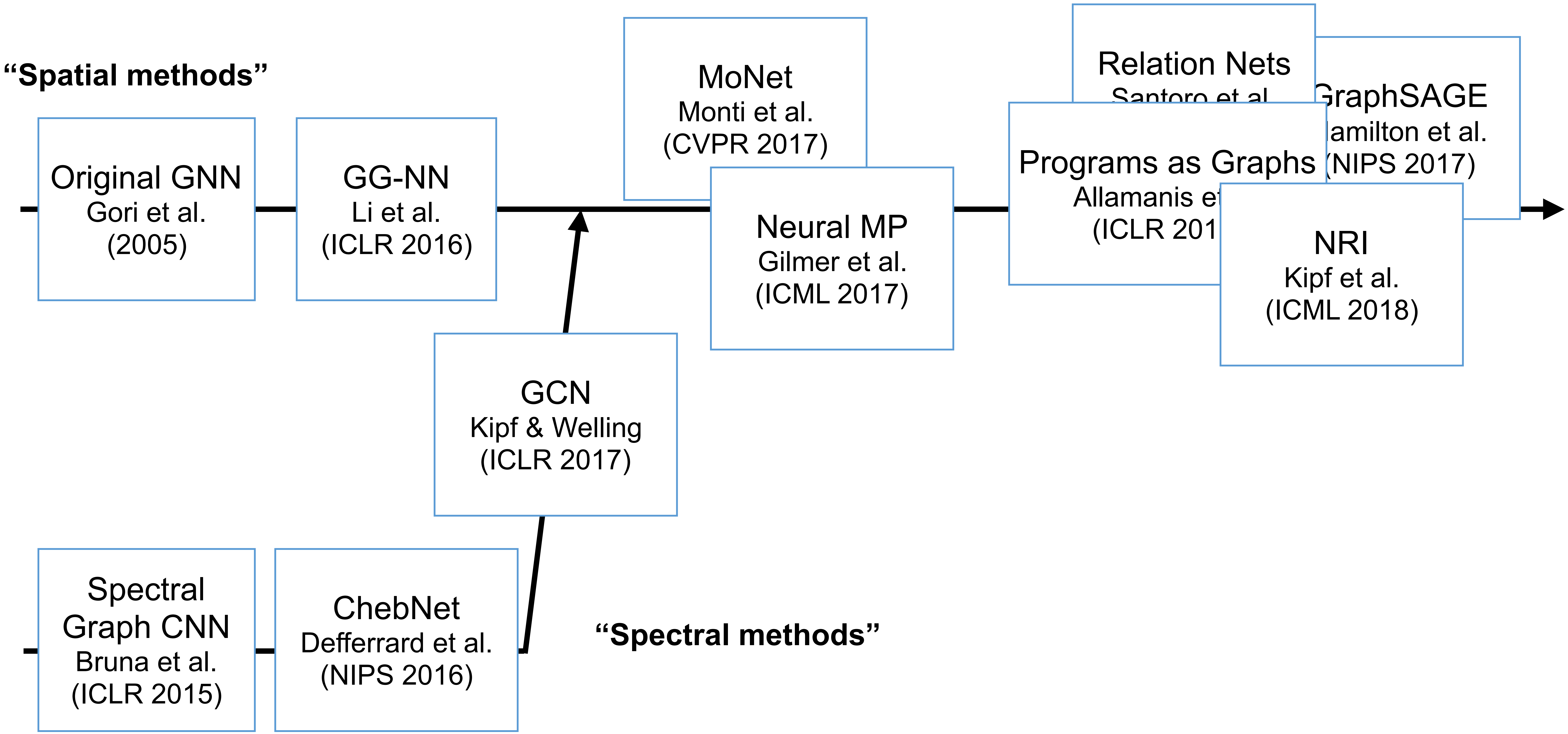
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



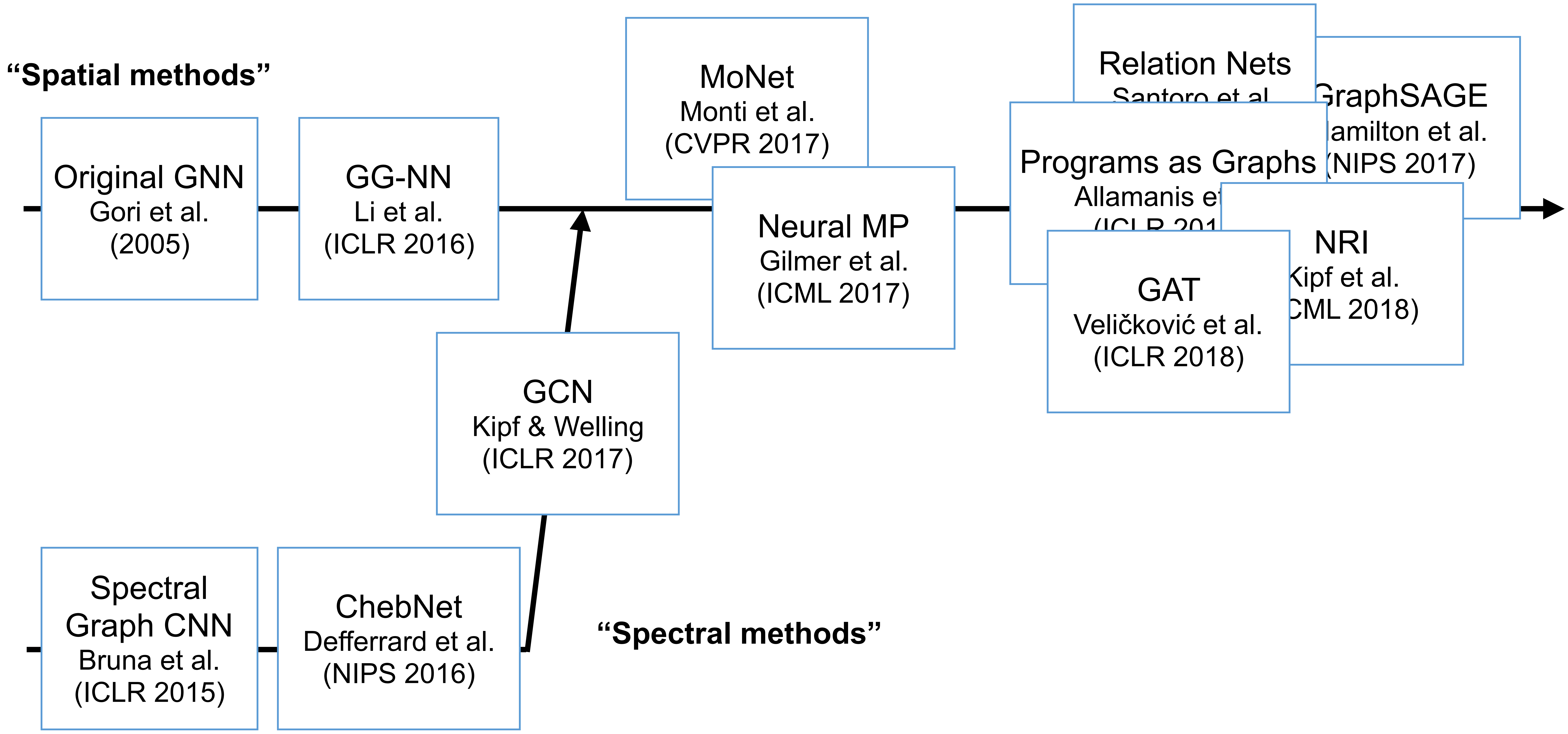
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



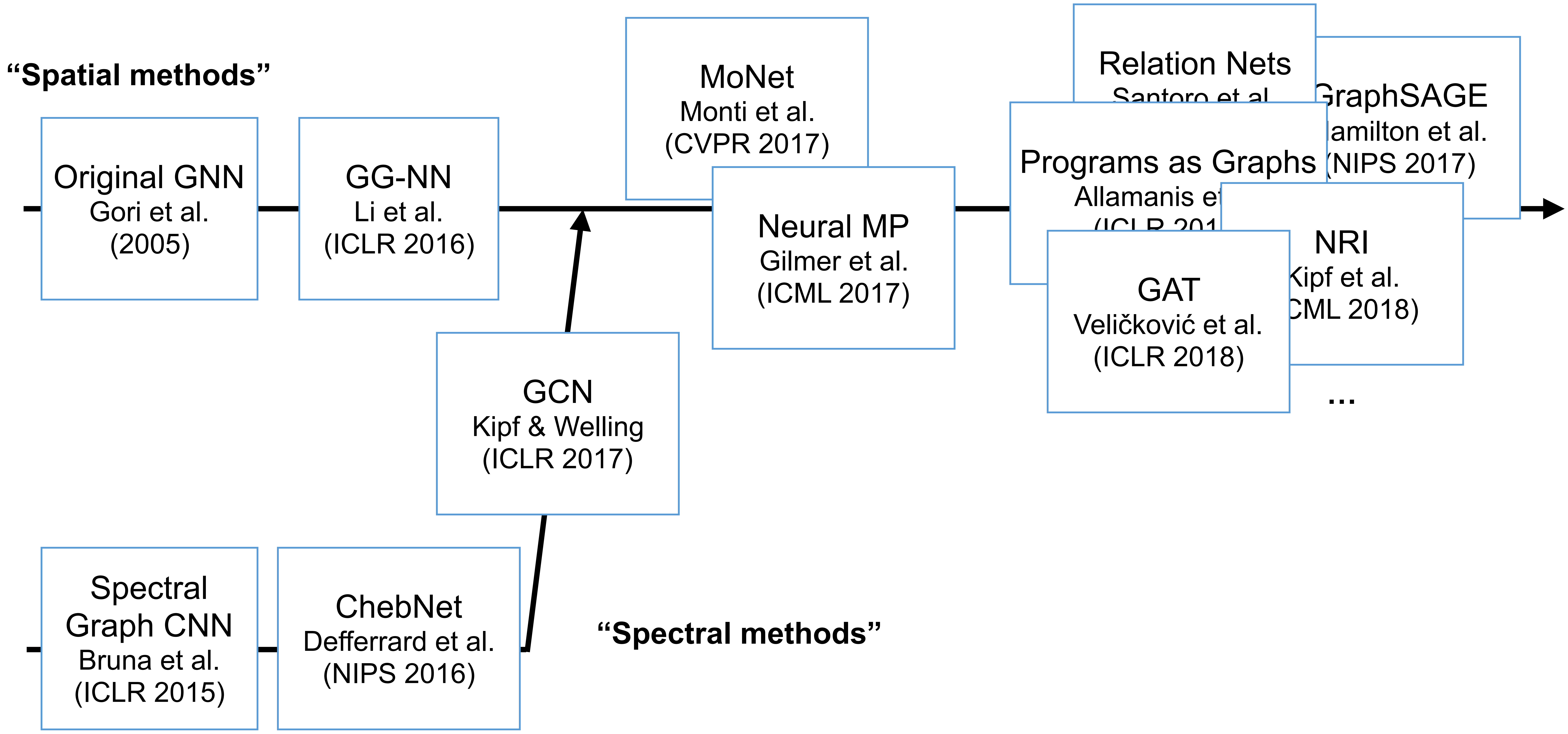
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



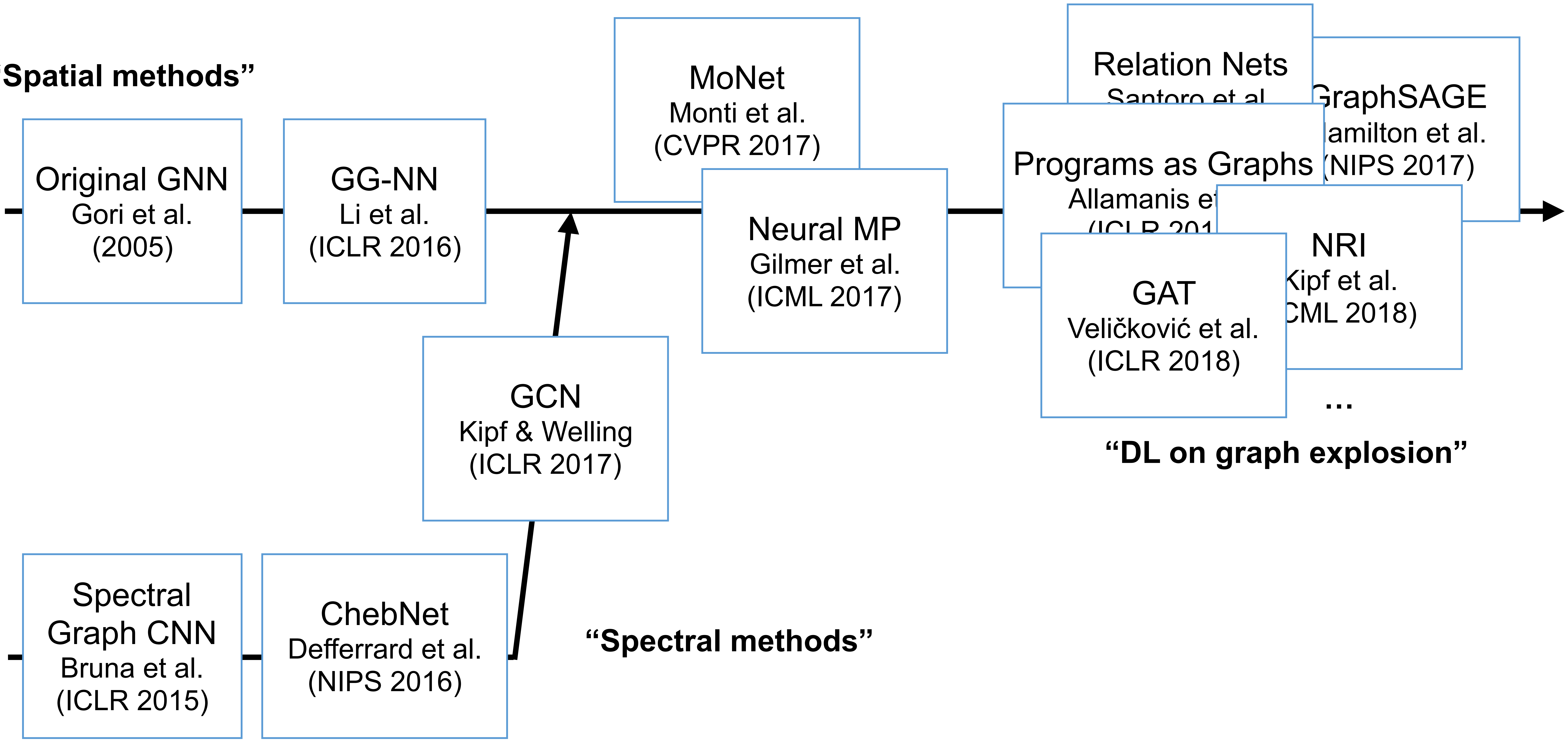
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



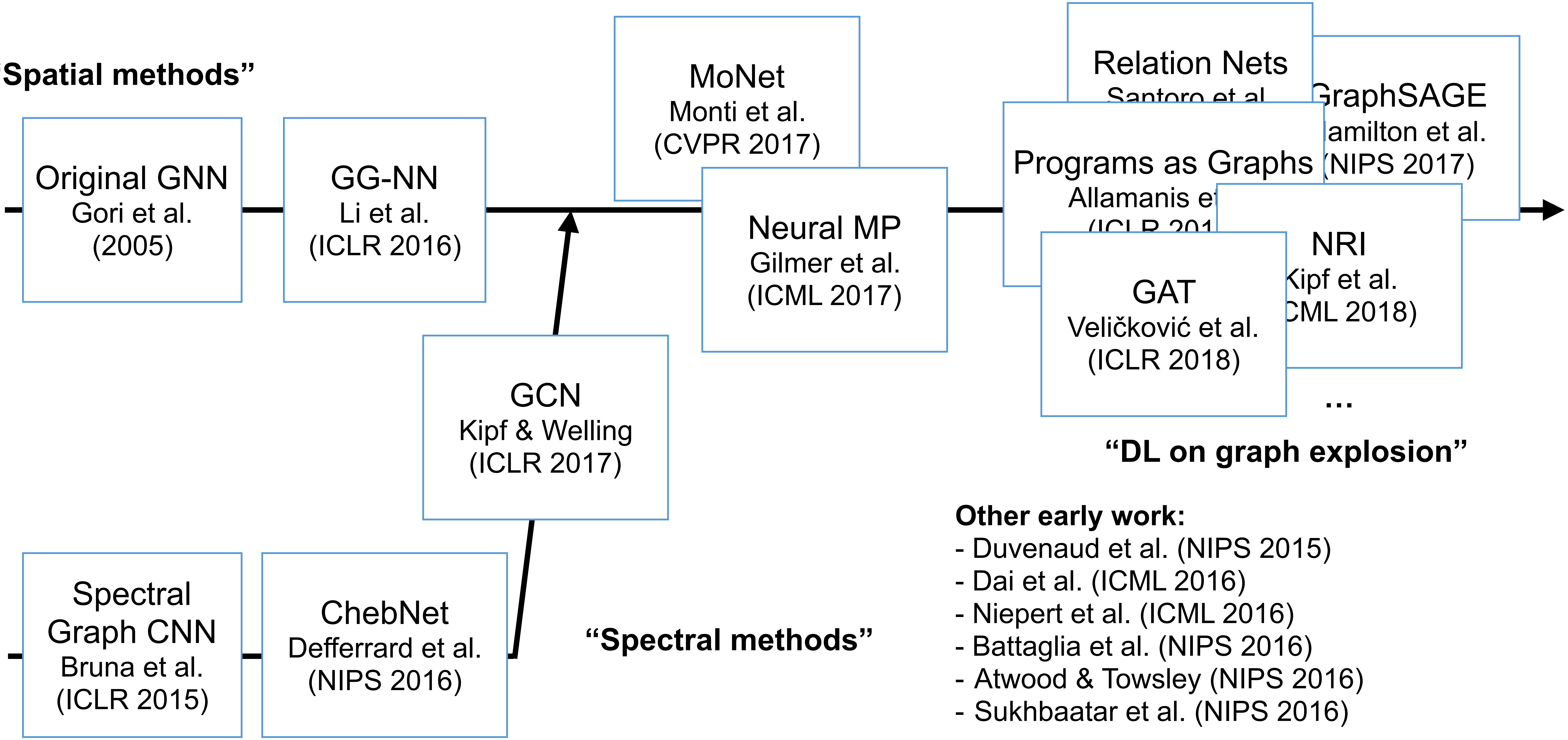
(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



(slide inspired by Alexander Gaunt's talk on GNNs)

A brief history of graph neural nets



Other early work:
- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt's talk on GNNs)

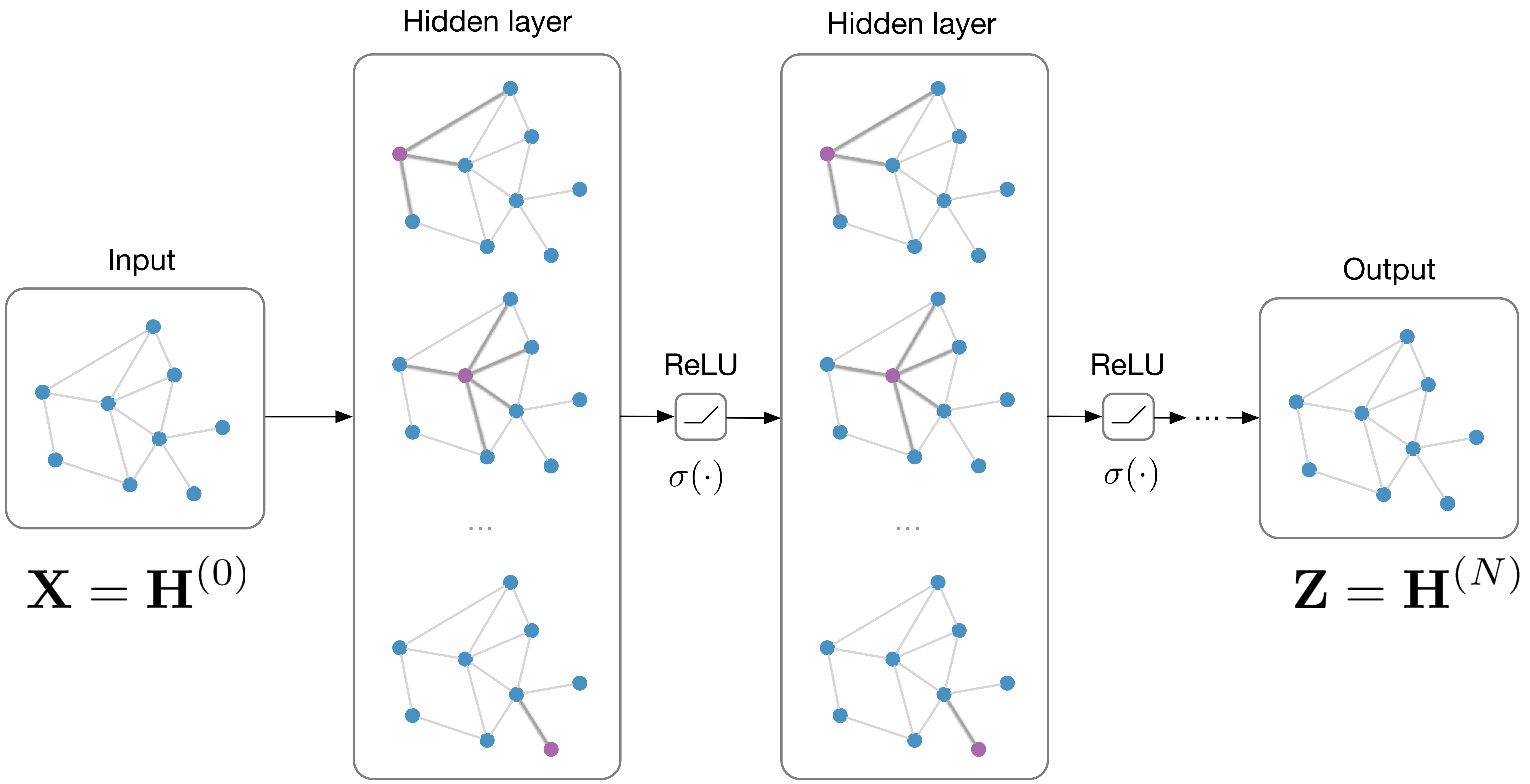
Part 2: Application to "classical" network problems

Node classification **Graph classification**

Link prediction

One fits all: Classification and link prediction with GNNs/GCNs

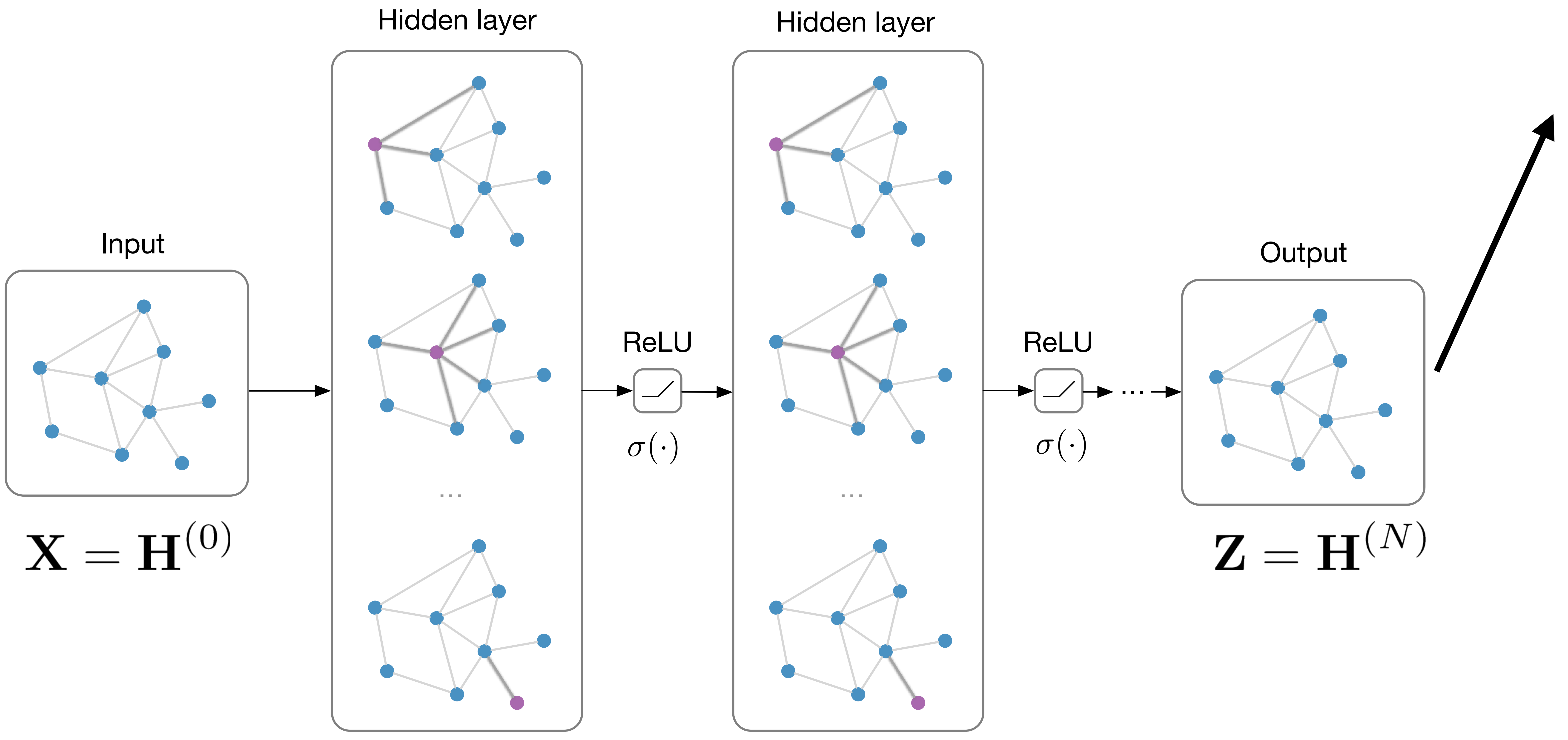
Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

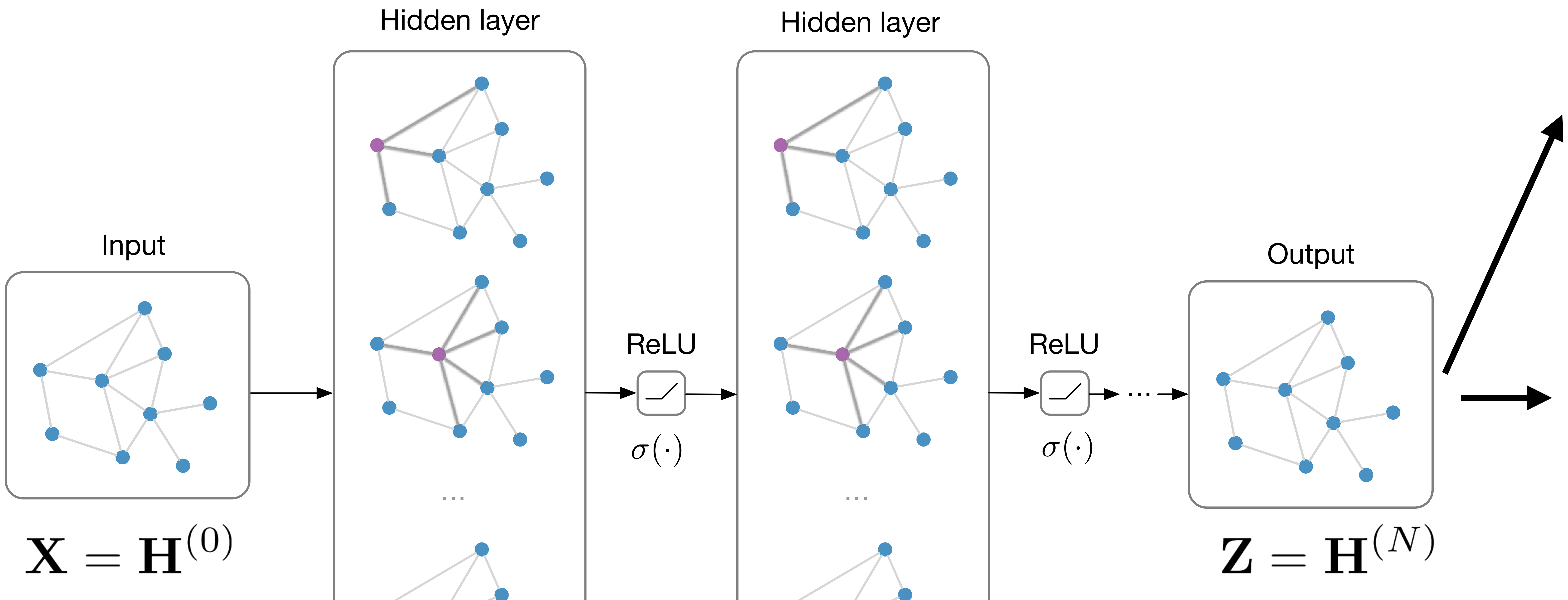
Node classification:

$$\mathrm{softmax}(\mathbf{z_n})$$

e.g. Kipf & Welling (ICLR 2017)

One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Node classification:

$$\mathrm{softmax}(\mathbf{z_n})$$

e.g. Kipf & Welling (ICLR 2017)

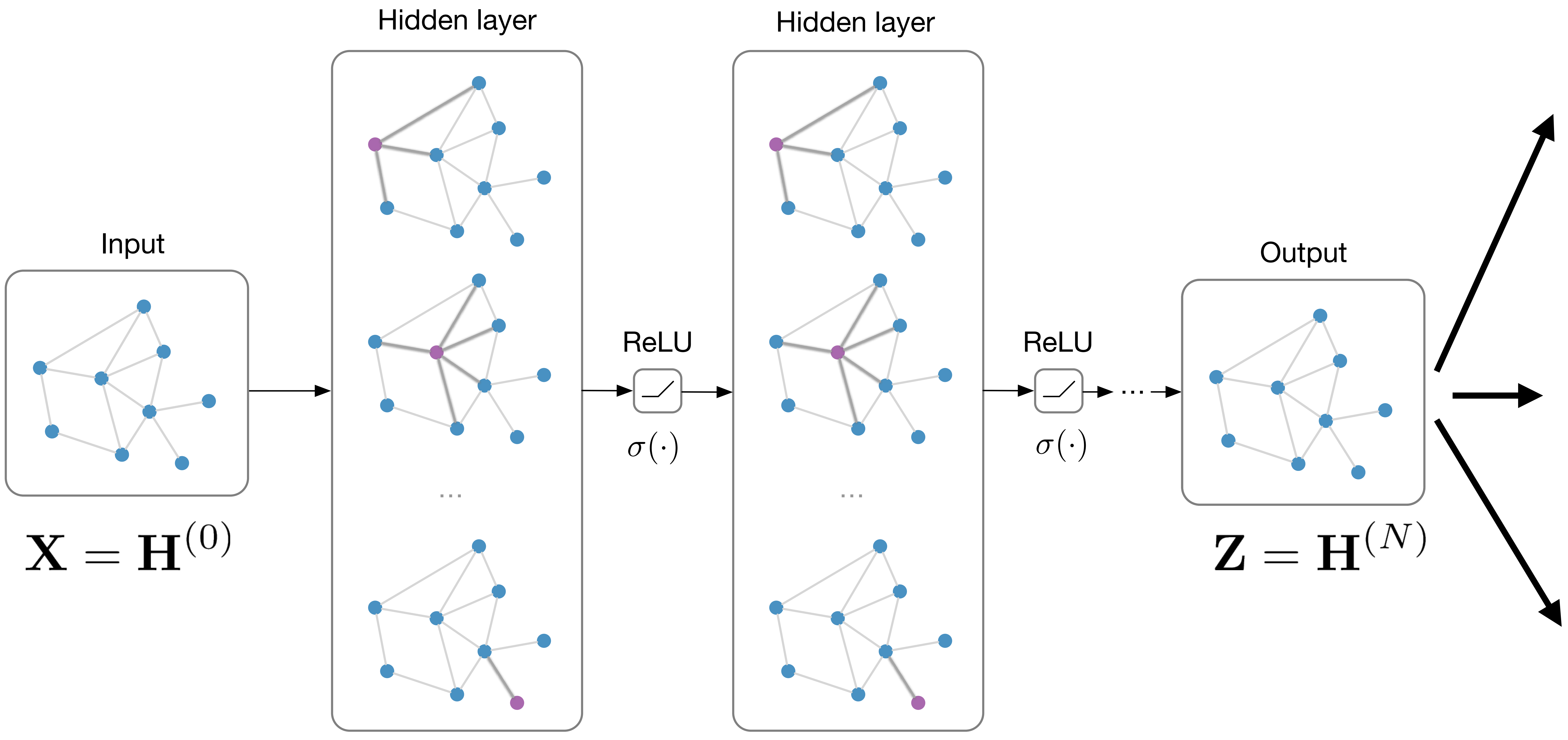
Graph classification:

$$\mathrm{softmax}(\textstyle\sum_n \mathbf{z_n})$$

e.g. Duvenaud et al. (NIPS 2015)

One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$\mathbf{X} = \mathbf{H}^{(0)}$

$\mathbf{Z} = \mathbf{H}^{(N)}$

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Node classification:

$$\mathrm{softmax}(\mathbf{z_n})$$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$$\mathrm{softmax}(\textstyle\sum_n \mathbf{z_n})$$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$$

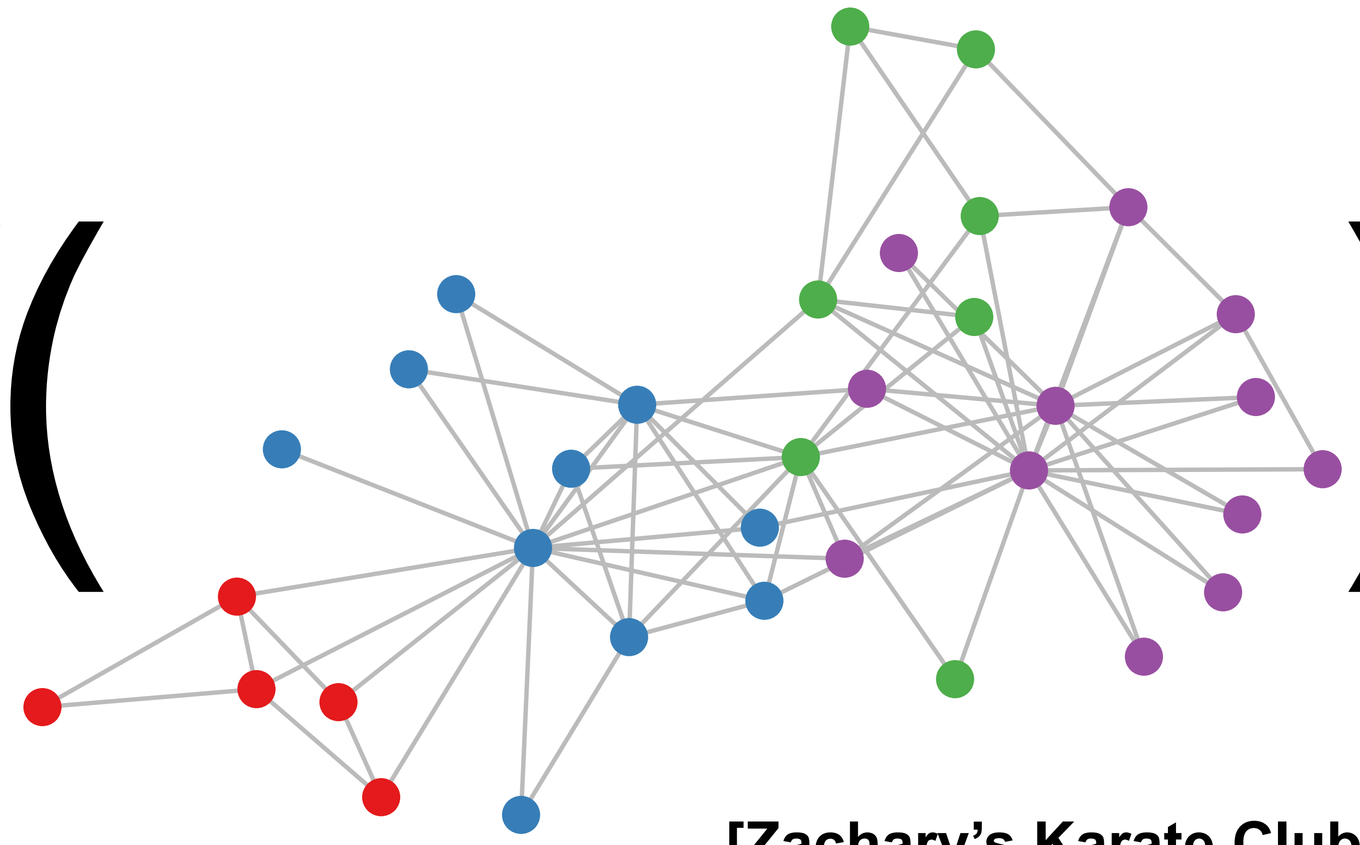
Kipf & Welling (NIPS BDL 2016)

"Graph Auto-Encoders"

What do learned representations look like?

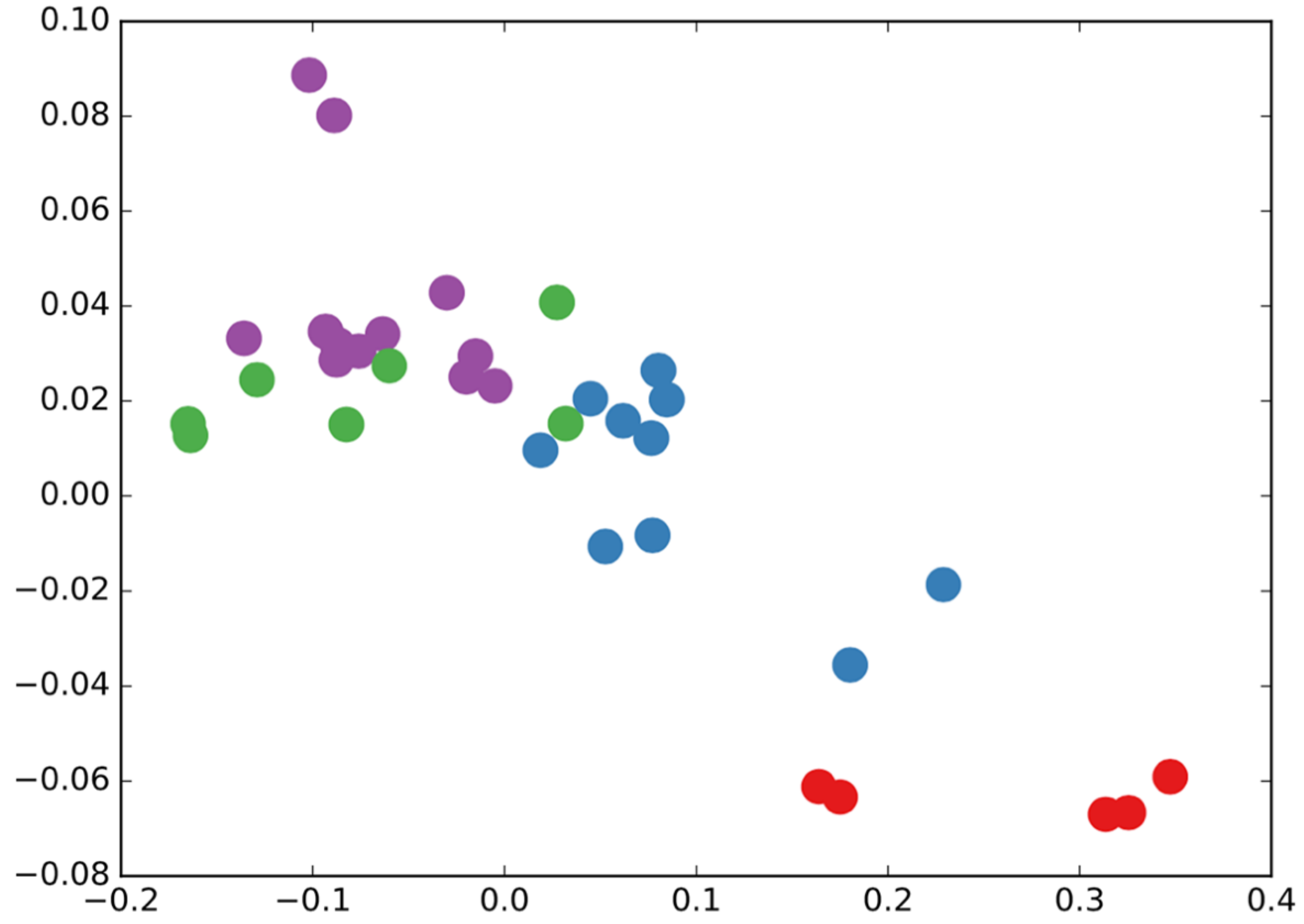
Forward pass through **untrained** 3-layer GCN model

Parameters initialized randomly

f() =

[Zachary's Karate Club]

2-dim output per node

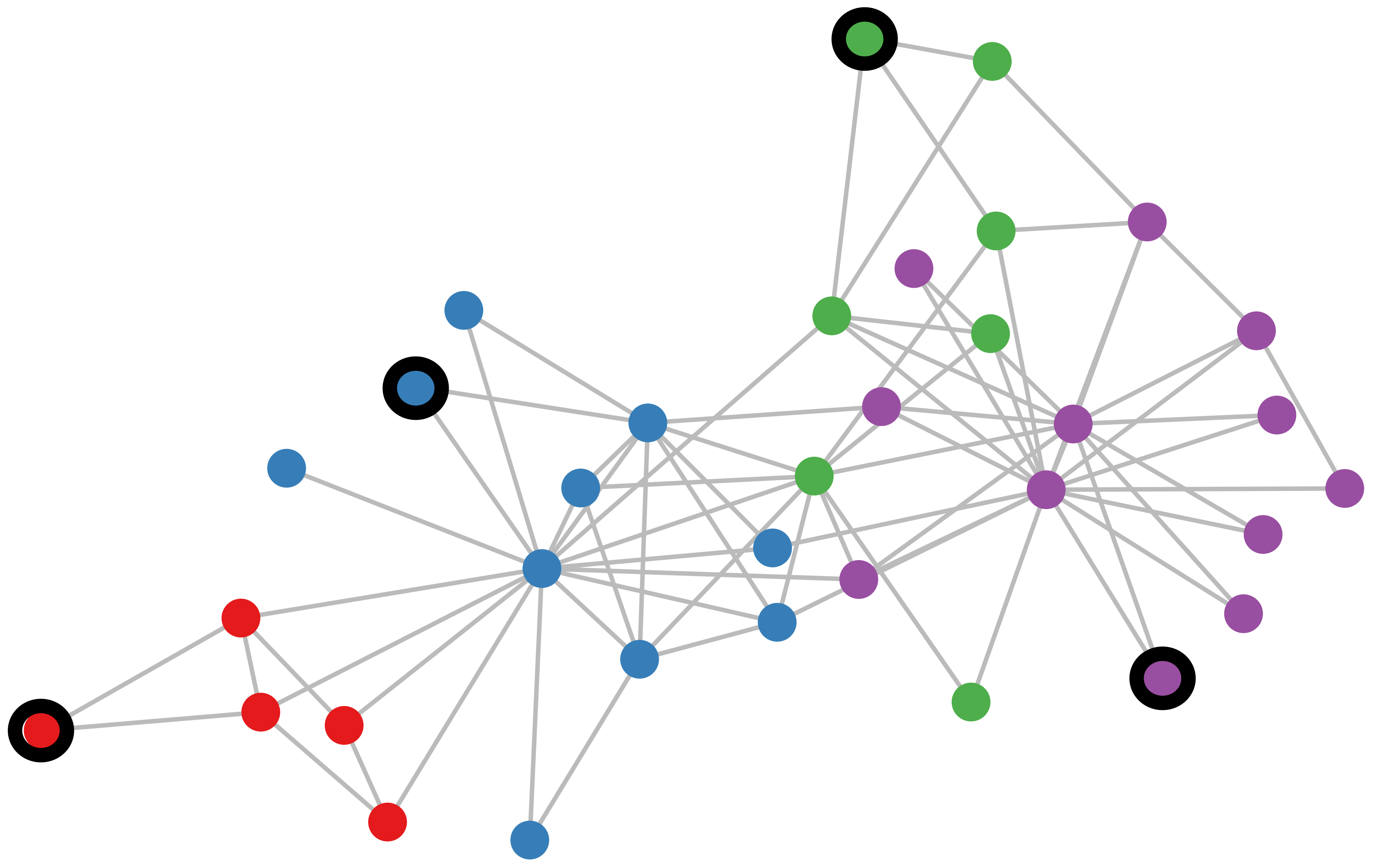


Semi-supervised classification on graphs

Setting:

Some nodes are labeled (black circle)
All other nodes are unlabeled

Task:
Predict node label of unlabeled nodes

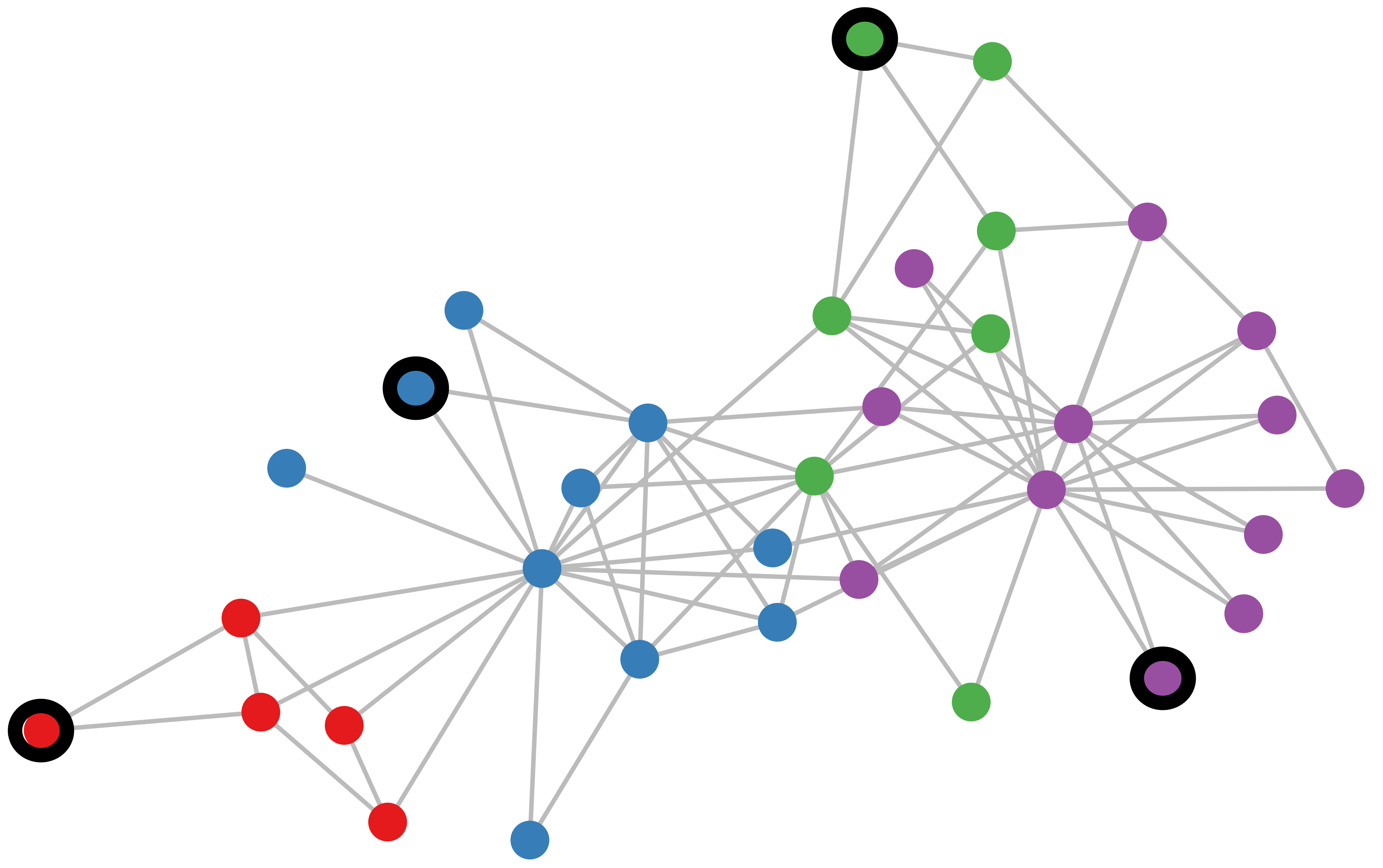


Semi-supervised classification on graphs

Setting:

Some nodes are labeled (black circle)
All other nodes are unlabeled

Task:
Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

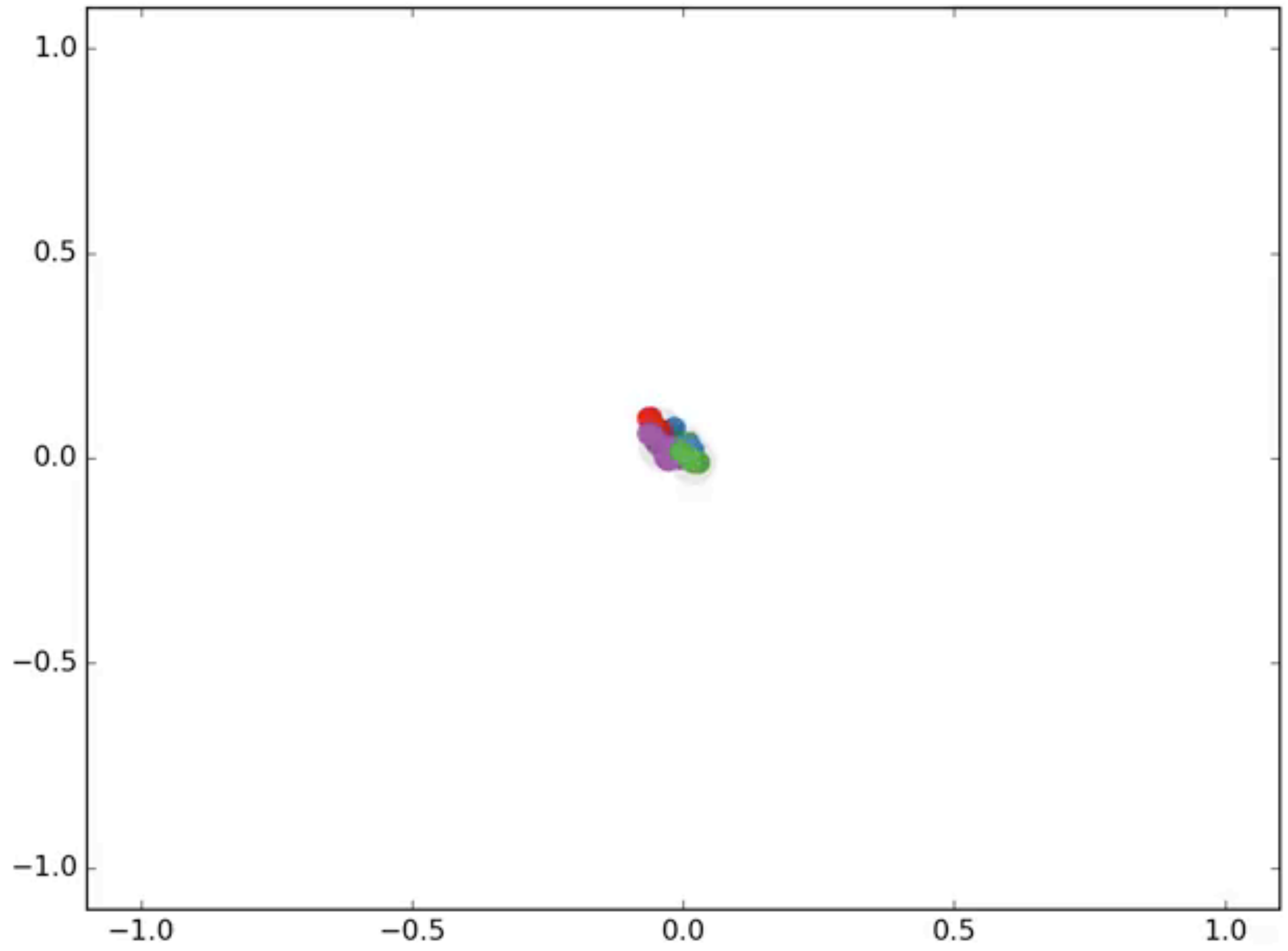
$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

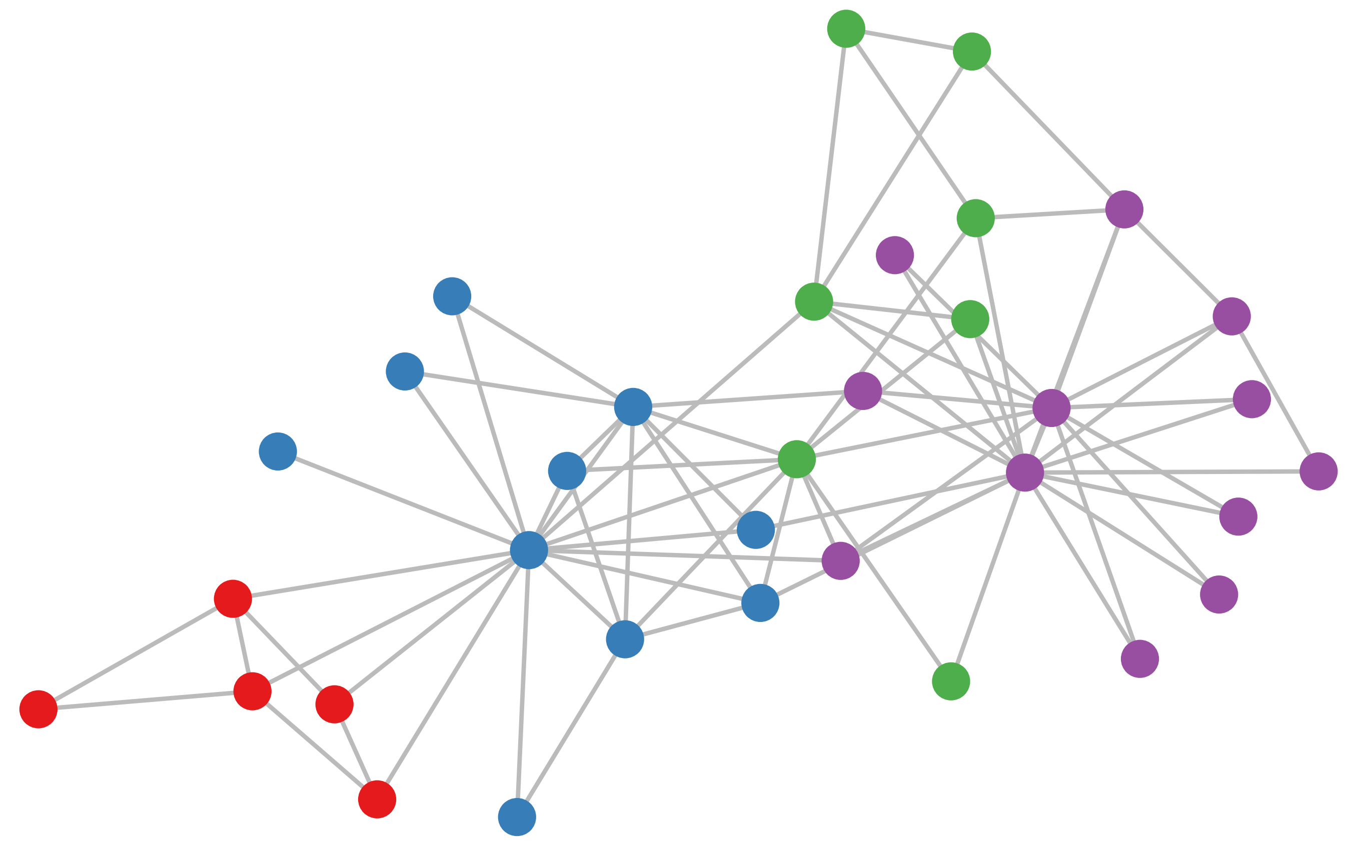
$\mathcal{Y}_L$ set of labeled node indices

$\mathbf{Y}$ label matrix

$\mathbf{Z}$ GCN output (after softmax)

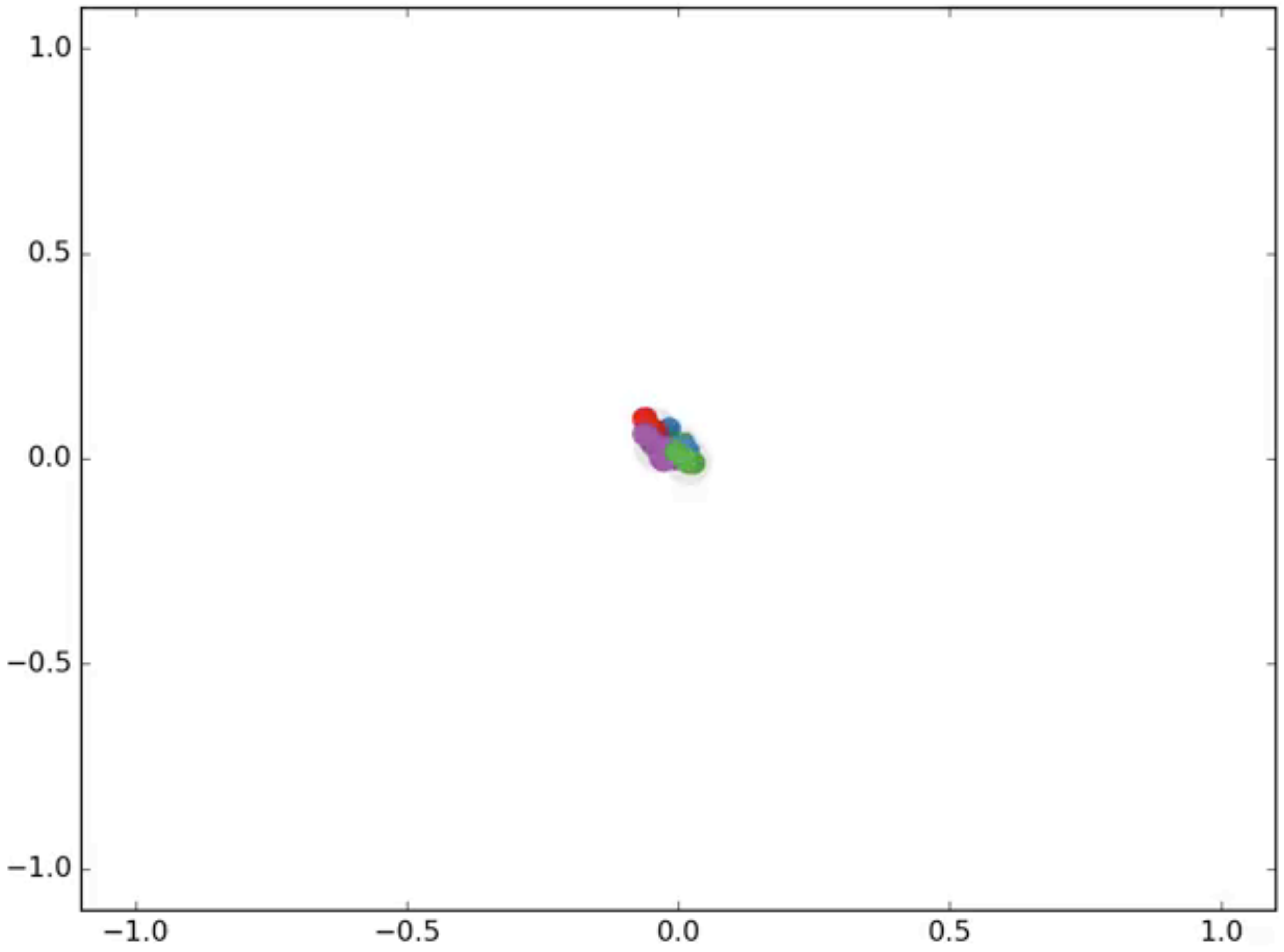
Toy example (semi-supervised learning)

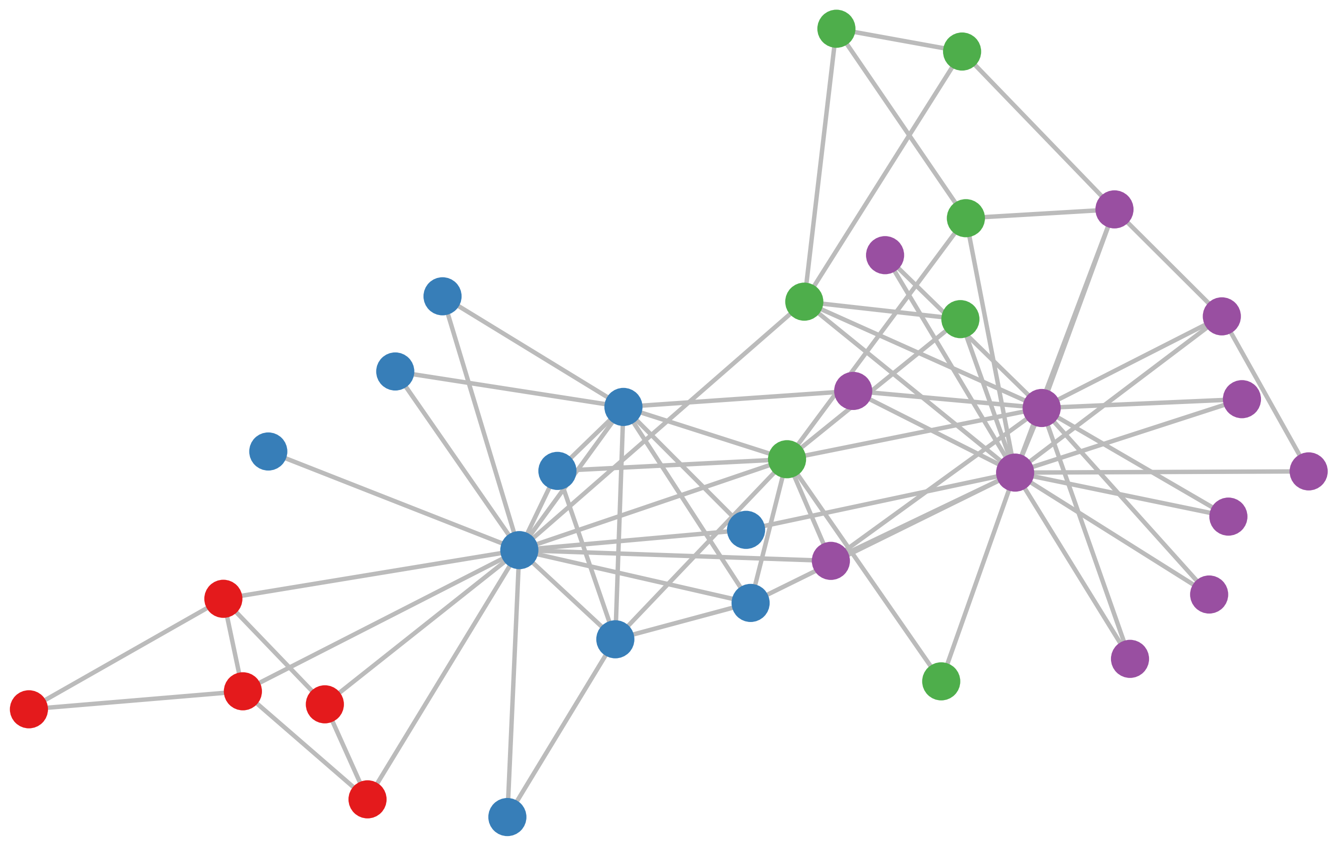




Video also available here:
http://tkipf.github.io/graph-convolutional-networks

Toy example (semi-supervised learning)





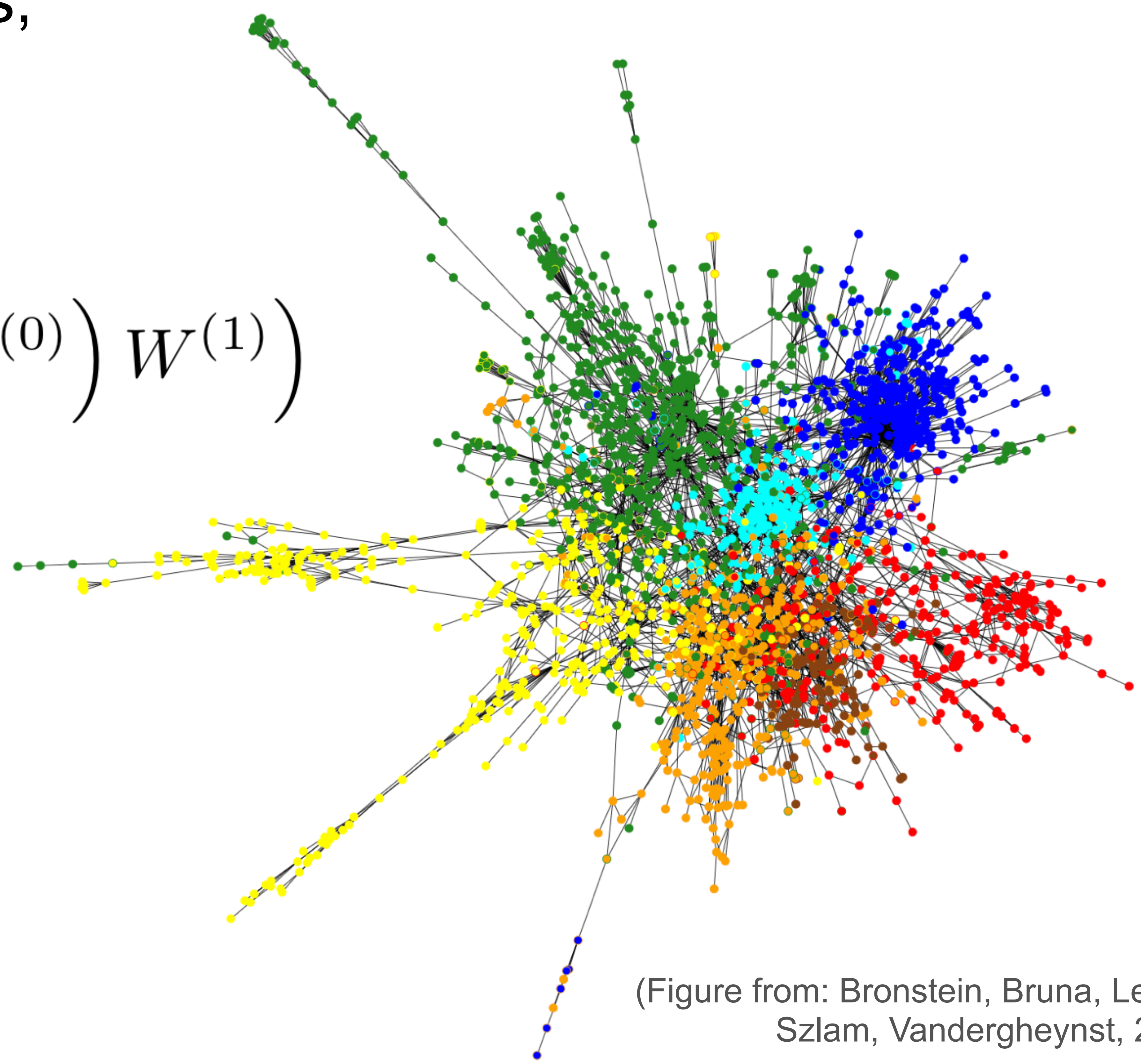
Video also available here:
http://tkipf.github.io/graph-convolutional-networks

Application: Classification on citation networks

Input: Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

Target: Paper category (e.g. stat.ML, cs.LG, …)

Model: 2-layer GCN $Z = f(X, A) = \text{softmax}\Big(\hat{A}\ \text{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big)$



(Figure from: Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2016)

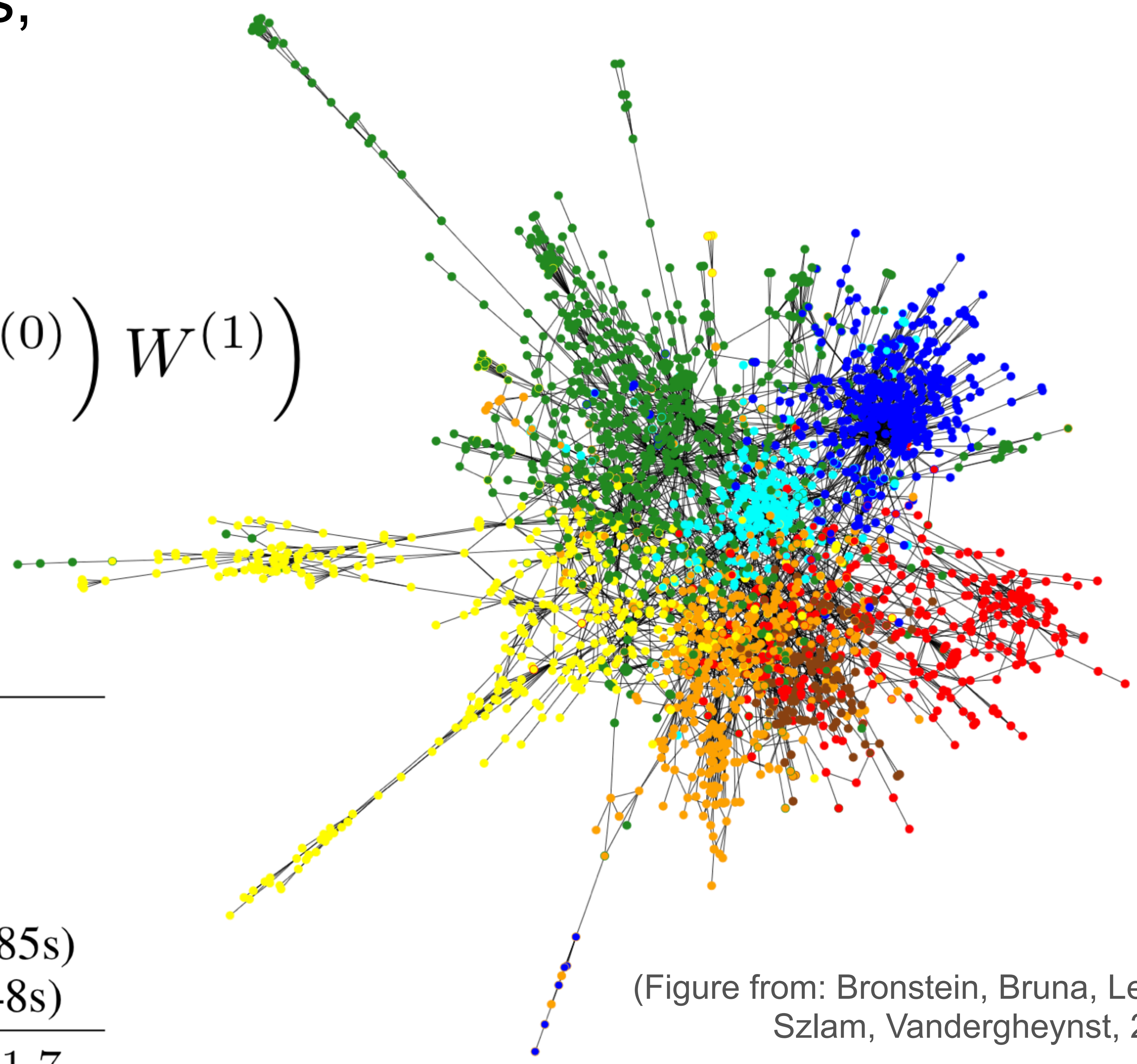
Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Application: Classification on citation networks

Input: Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

Target: Paper category (e.g. stat.ML, cs.LG, …)

Model: 2-layer GCN $Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big)$



(Figure from: Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2016)

Classification results (accuracy)

no input features → LP [27], DeepWalk [18]

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [24]	59.6	59.0	71.1	26.7
LP [27]	45.3	68.0	63.0	26.5
DeepWalk [18]	43.2	67.2	65.3	58.1
Planetoid* [25]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Part 3: Emerging research directions for structured deep models

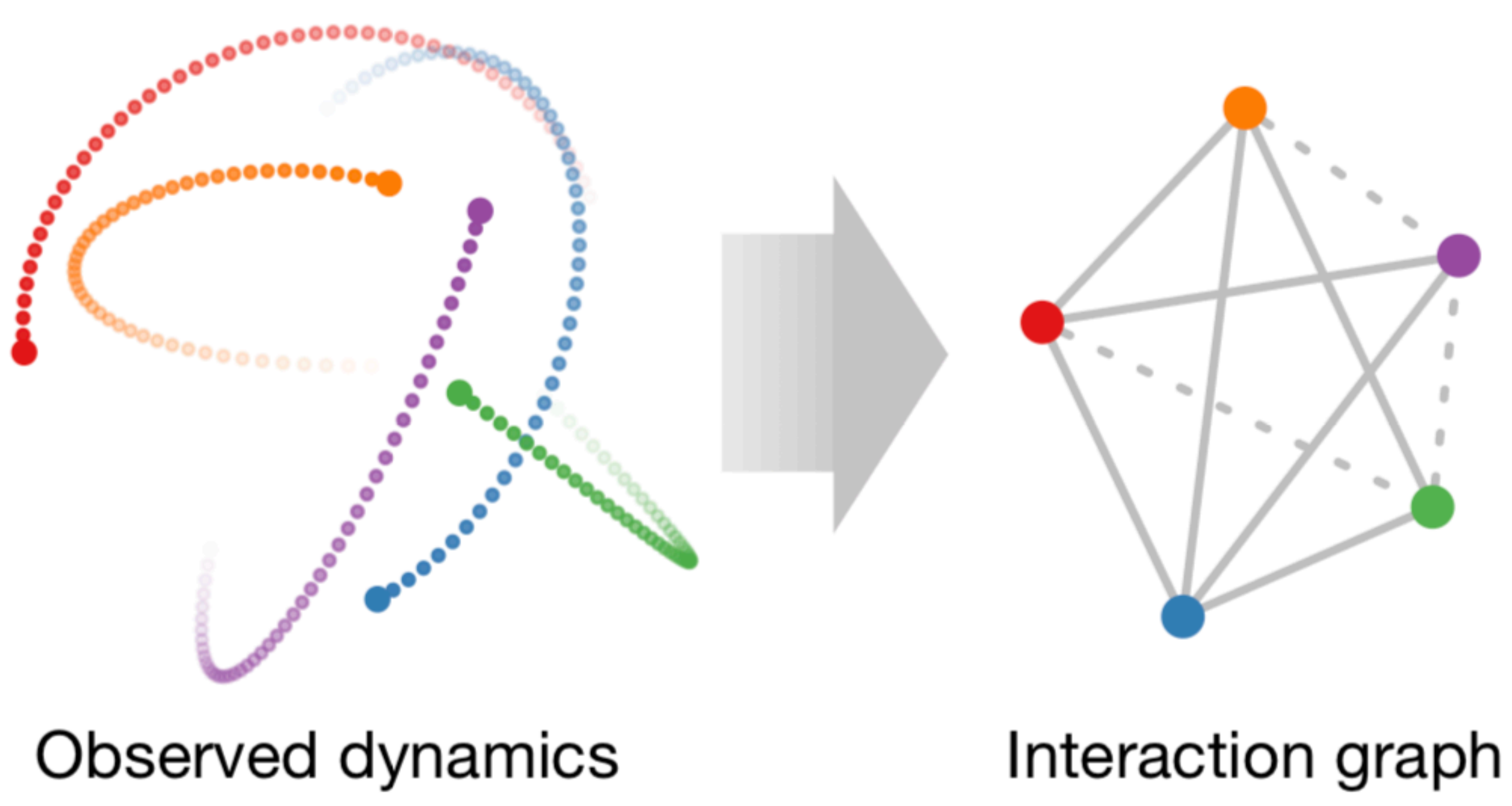
Latent graph inference

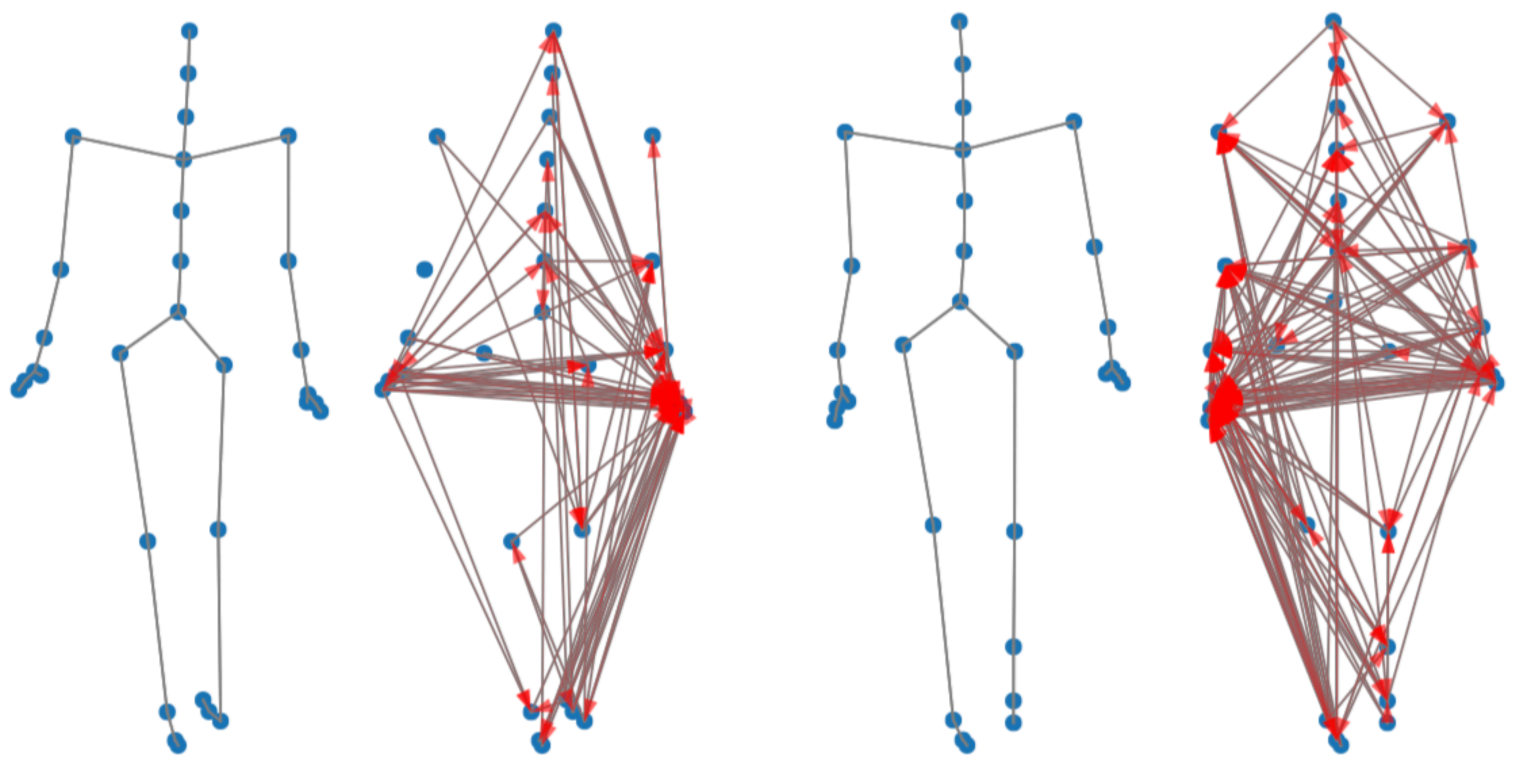
Deep generative models for graphs

Latent graph inference

Neural Relational Inference for Interacting Systems

Thomas Kipf [*1] **Ethan Fetaya** [*2,3] **Kuan-Chieh Wang** [2,3] **Max Welling** [1,4] **Richard Zemel** [2,3,4]





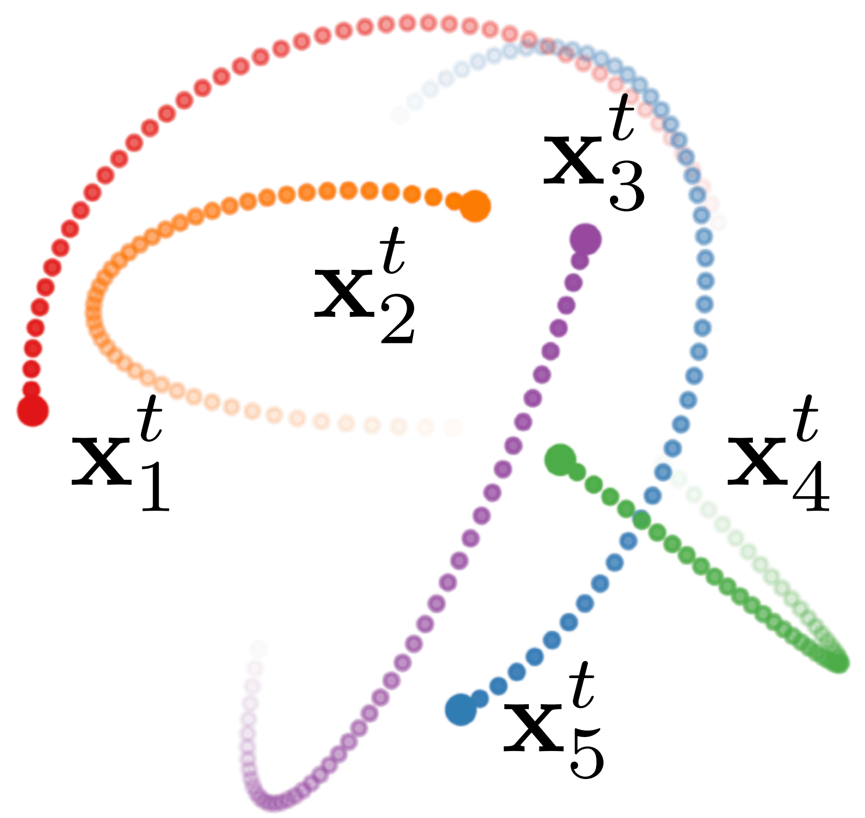
(a) Right hand focus (b) Left hand focus

To be presented at ICML 2018!

Motivation: Learning physical dynamics

Need to model **interactions** and their **effect on dynamics**

Simple example:



Observed dynamics

- 5 particles + their trajectories $\mathbf{x}_i^t$
- Concatenate feature vectors

$$\mathbf{x}^t = [\mathbf{x}_1^t, \mathbf{x}_2^t, ..., \mathbf{x}_5^t]$$

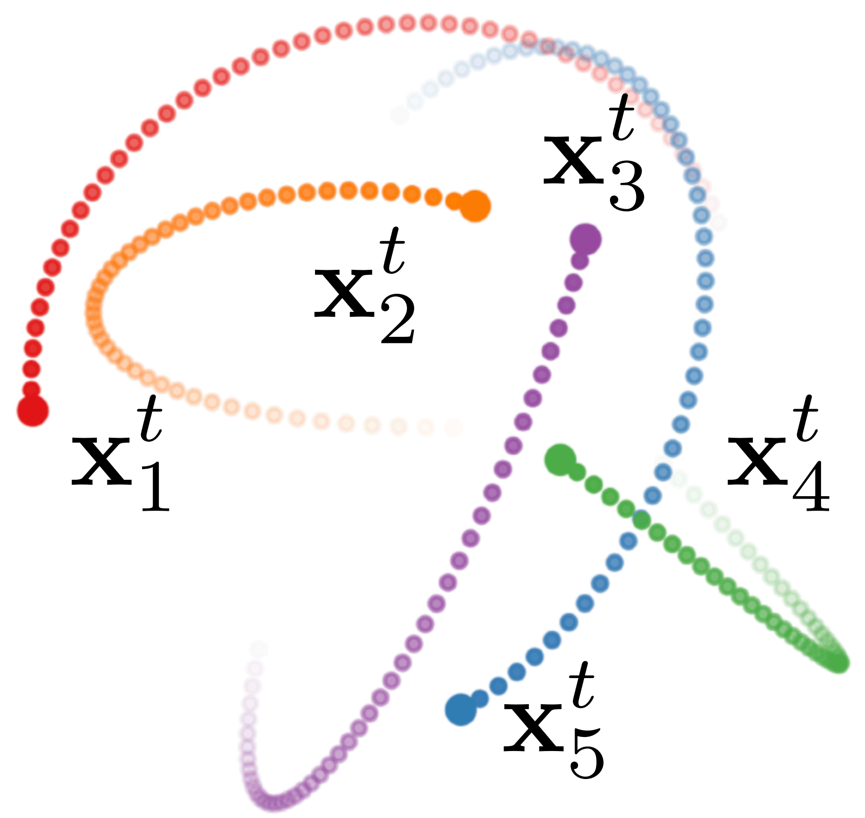
- Feed into neural net

$$\mathbf{x}^{t+1} = \mathrm{MLP}(\mathbf{x}^t) \quad \text{(or RNN)}$$

Motivation: Learning physical dynamics

Need to model **interactions** and their **effect on dynamics**

Simple example:



Observed dynamics

- 5 particles + their trajectories $\mathbf{x}_i^t$
- Concatenate feature vectors

$$\mathbf{x}^t = [\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_5^t]$$

- Feed into neural net

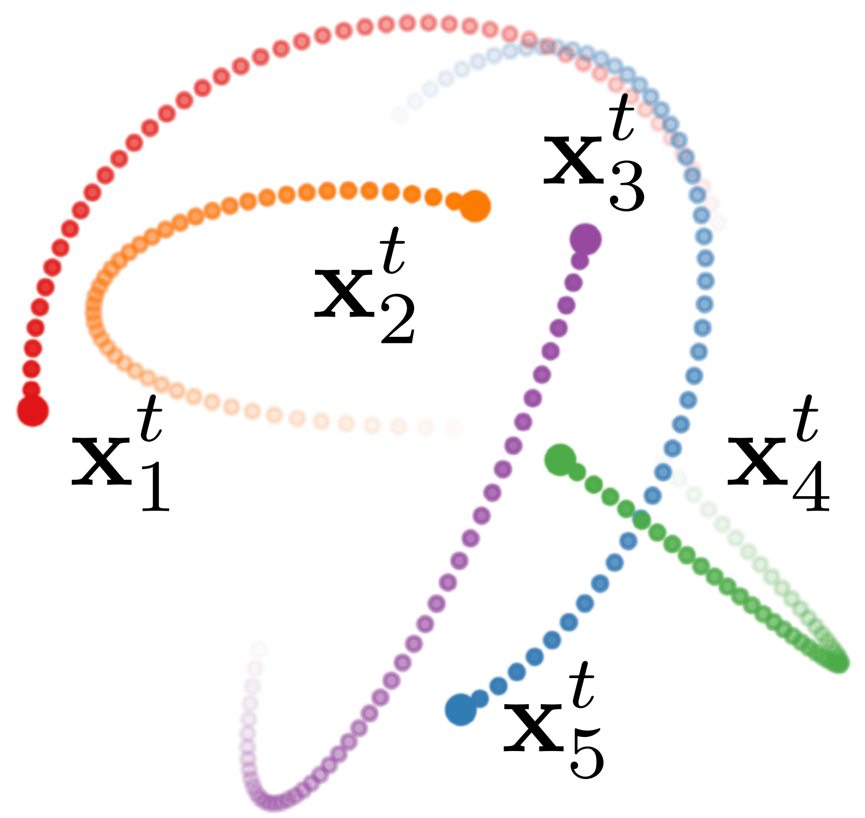
$$\mathbf{x}^{t+1} = \mathrm{MLP}(\mathbf{x}^t) \quad \text{(or RNN)}$$

- Done?

Motivation: Learning physical dynamics

Need to model **interactions** and their **effect on dynamics**

Simple example:



Observed dynamics

- 5 particles + their trajectories $\mathbf{x}_i^t$
- Concatenate feature vectors

$$\mathbf{x}^t = [\mathbf{x}_1^t, \mathbf{x}_2^t, ..., \mathbf{x}_5^t]$$

- Feed into neural net

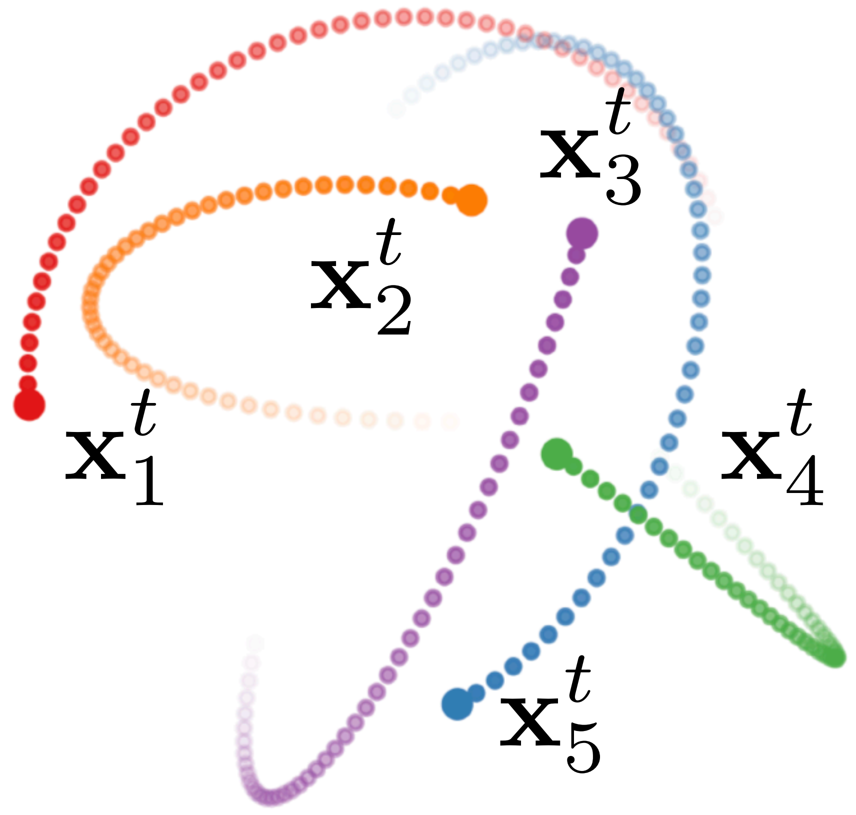
$$\mathbf{x}^{t+1} = \mathrm{MLP}(\mathbf{x}^t) \quad \text{(or RNN)}$$

- Done?

Works (somewhat), but we can do a lot better.

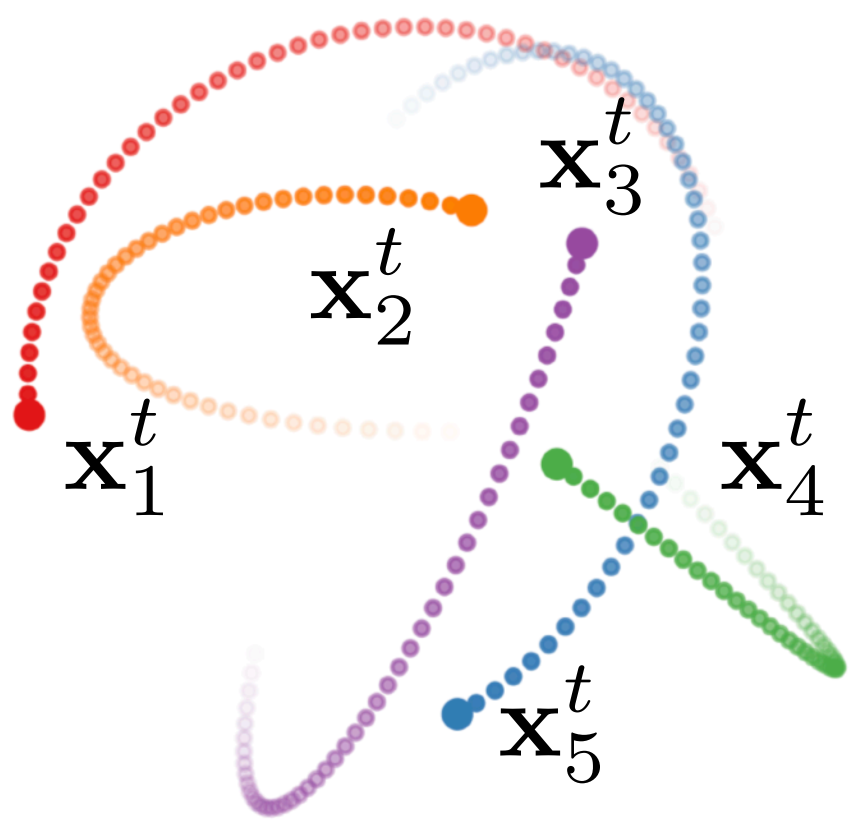
A naïve model of Intuitive Physics

Problems:



Observed dynamics

A naïve model of Intuitive Physics



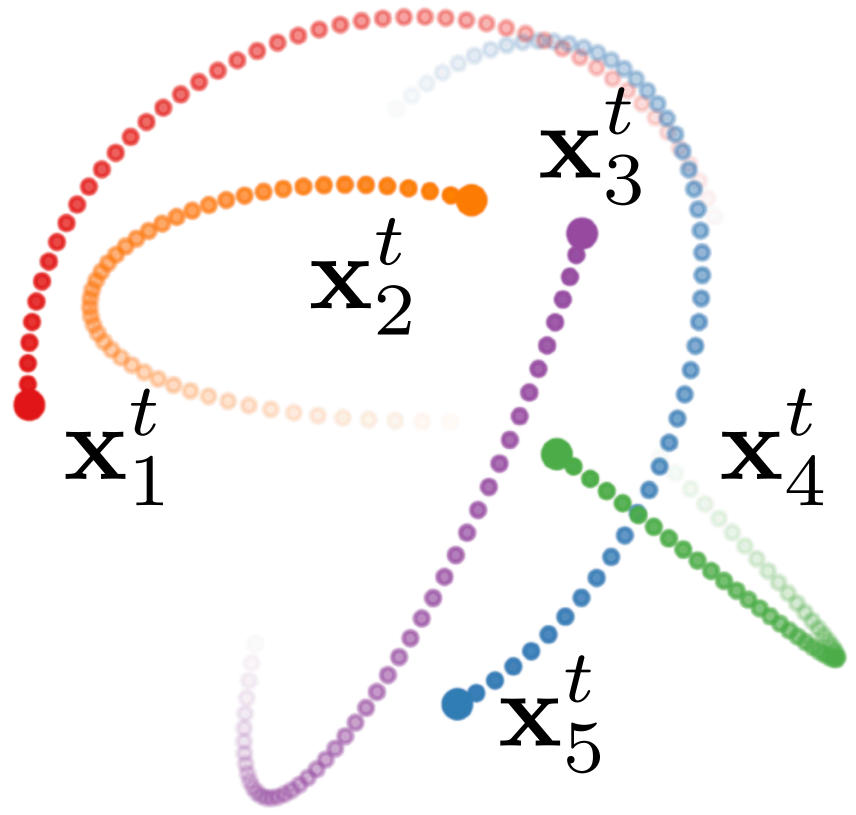
Observed dynamics

Problems:

- Arbitrary ordering of nodes

➡ Need permutation equivariance

A naïve model of Intuitive Physics

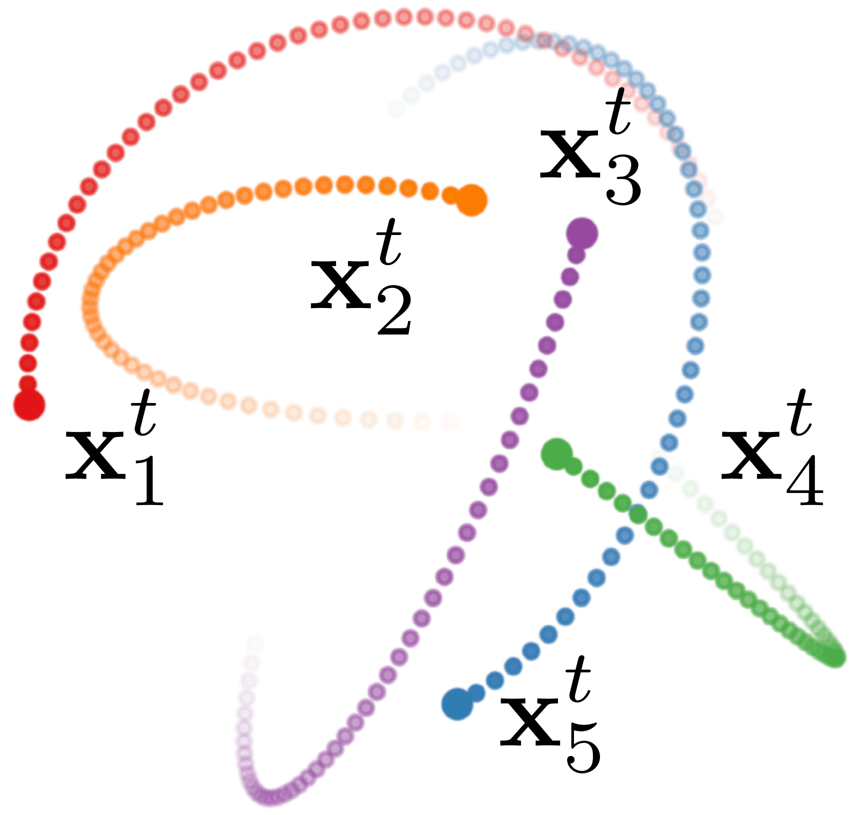


Observed dynamics

Problems:

- Arbitrary ordering of nodes
  ➡ Need permutation equivariance
- Model doesn't know about **structure** of interactions
  ➡ For many fundamental physical systems, interactions are **pairwise**

A naïve model of Intuitive Physics



Observed dynamics

Problems:

- Arbitrary ordering of nodes

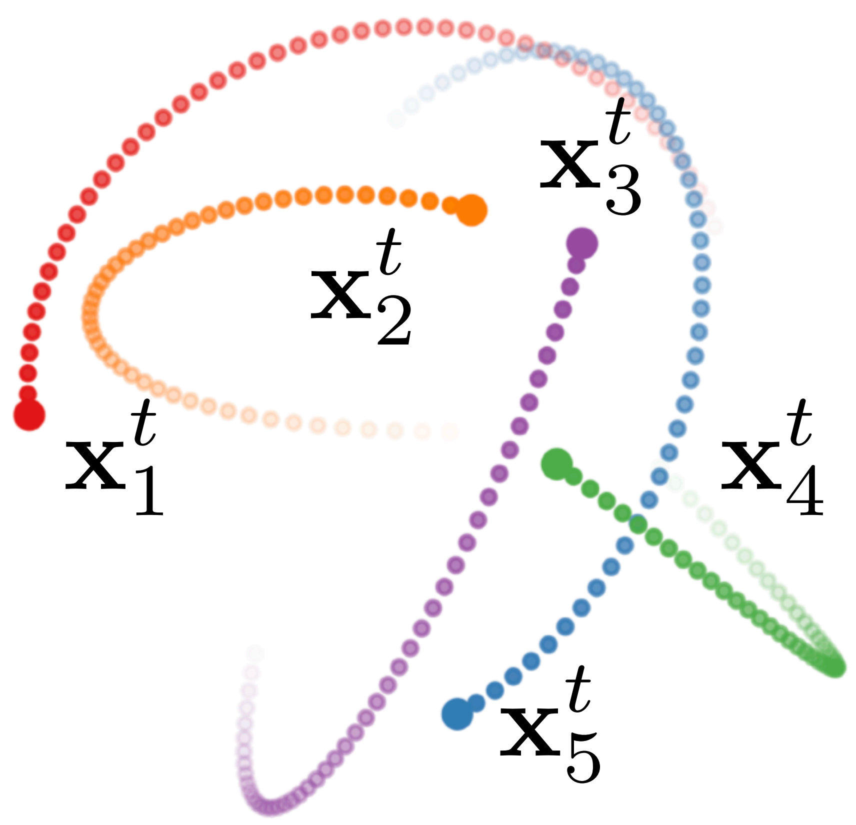
  ➡ Need permutation equivariance

- Model doesn't know about **structure** of interactions

  ➡ For many fundamental physical systems, interactions are **pairwise**

<u>Structured</u> neural models to the rescue!

A naïve model of Intuitive Physics



Observed dynamics

Problems:

- Arbitrary ordering of nodes
  ➡ Need permutation equivariance
- Model doesn't know about **structure** of interactions
  ➡ For many fundamental physical systems, interactions are **pairwise**

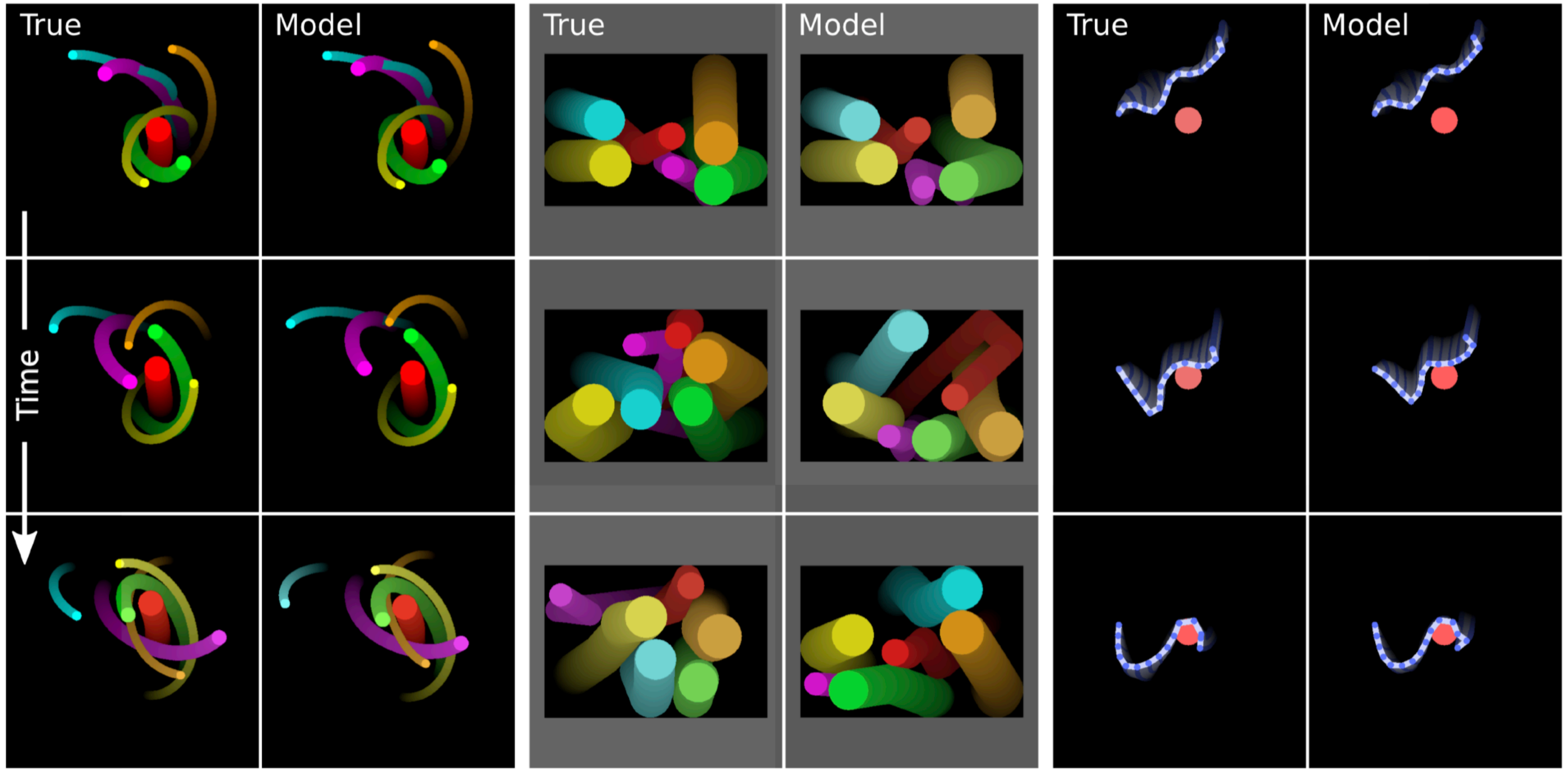
<u>Structured</u> neural models to the rescue!

Graph Neural Networks (GNNs) are an ideal candidate.

GNNs for interacting systems

Using GNNs, we can learn to model physical dynamics of interacting systems with very high precision

<u>**if**</u> we know about the underlying **structure** of the interactions and their **types** (should there be different types)



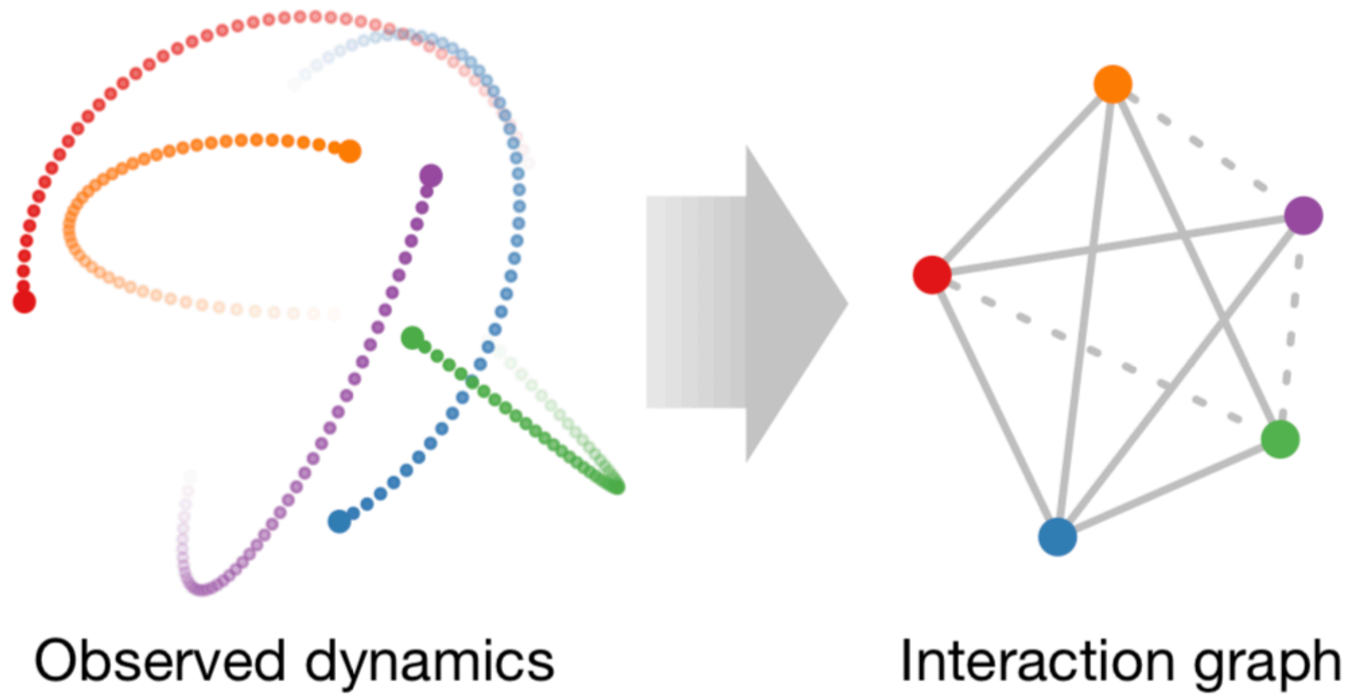
Battaglia et al., (NIPS 2016)

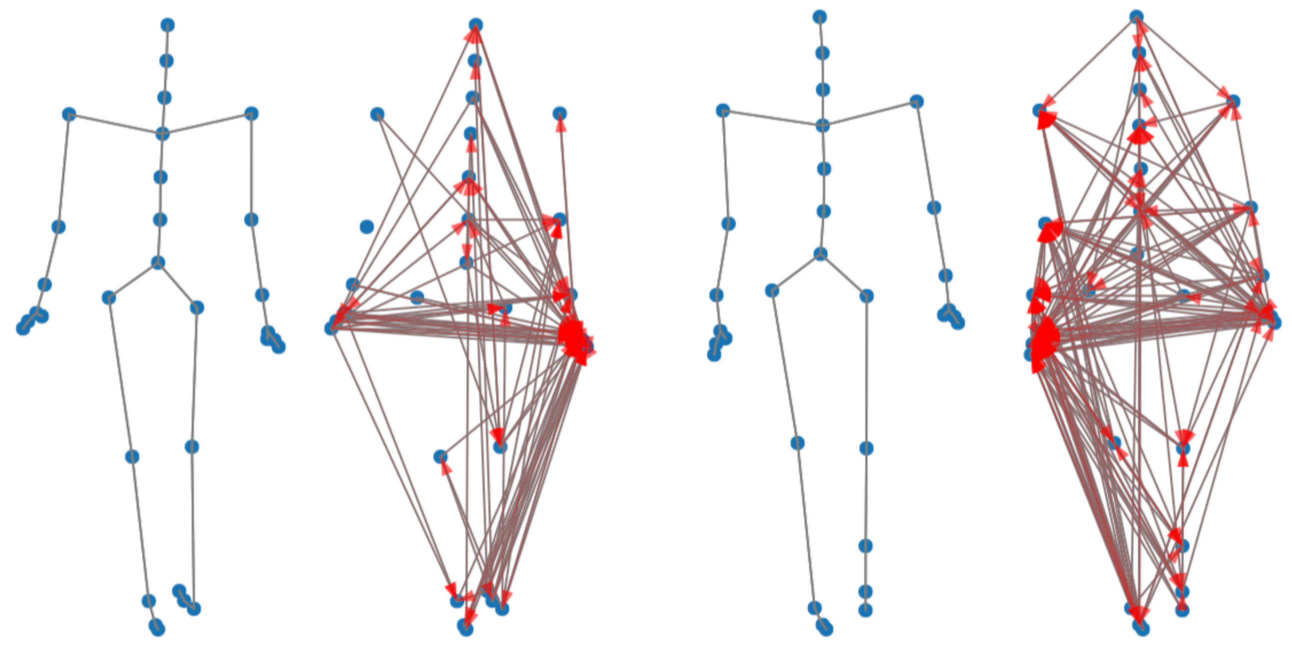
Neural Relational Inference for Interacting Systems

Thomas Kipf[*1] **Ethan Fetaya**[*2 3] **Kuan-Chieh Wang**[2 3] **Max Welling**[1 4] **Richard Zemel**[2 3 4]

Our work (ICML 2018)**:**

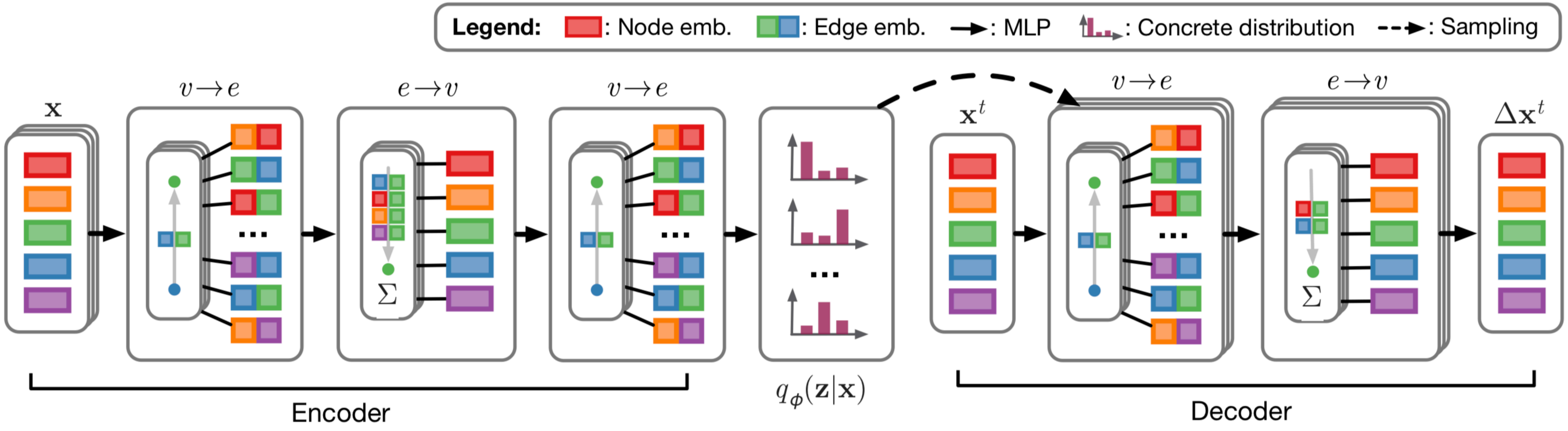
1. Learn dynamics of interacting system **without** knowing structure of interactions
2. Infer **latent interaction graph** (plus edge types) using a VAE
3. Applications for **physical systems**, **motion capture data** and **multi-agent systems**





(a) Right hand focus (b) Left hand focus

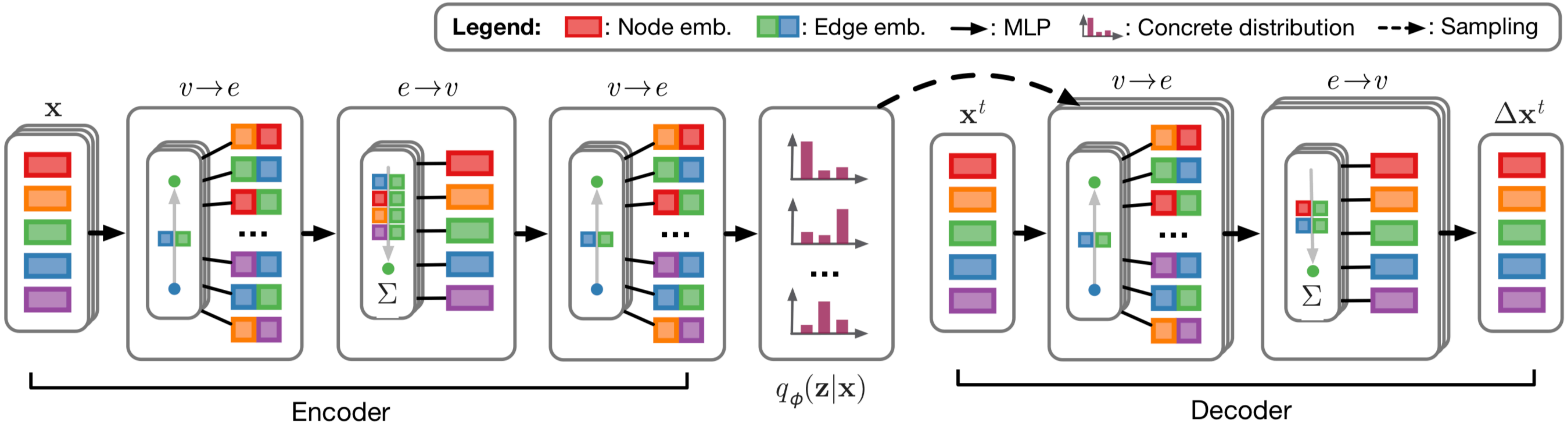
Neural Relational Inference - Model overview



Model: Variational auto-encoder with (discrete) edge types as discrete latent variables

Encoder and decoder are GNN-based!

Neural Relational Inference - Model overview



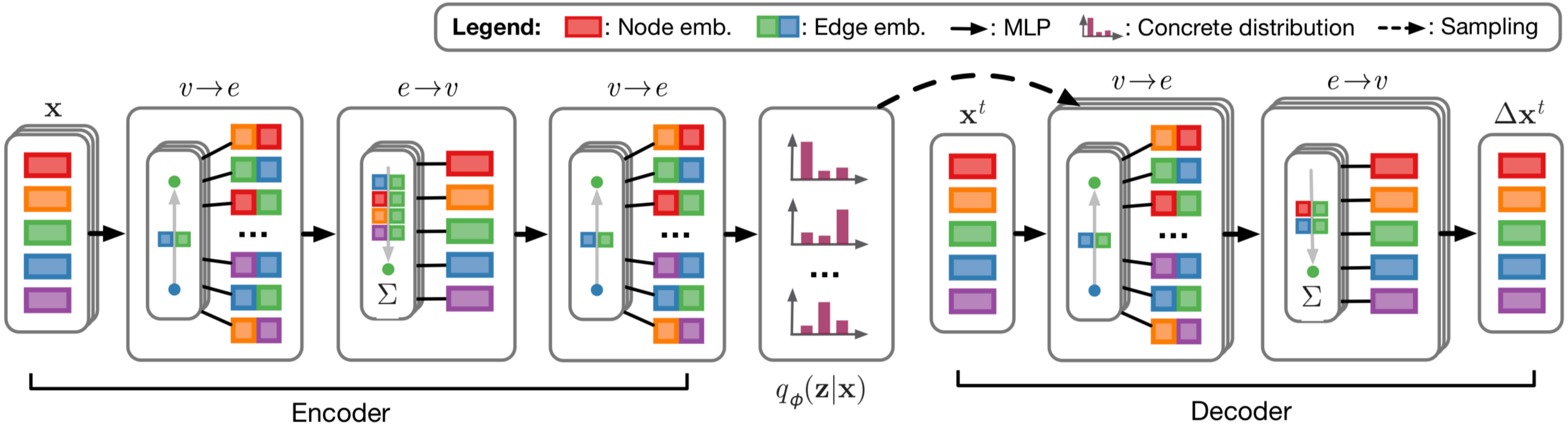
Model: Variational auto-encoder with (discrete) edge types as discrete latent variables

Encoder and decoder are GNN-based!

Main intuition:

- Encoder **generates hypothesis** on how the system interacts
- Decoder **learns a dynamical model** constrained by the encoder's "interaction hypothesis"

Neural Relational Inference - Model overview



Model: Variational auto-encoder with (discrete) edge types as discrete latent variables

Encoder and decoder are GNN-based!

Main intuition:

- Encoder **generates hypothesis** on how the system interacts
- Decoder **learns a dynamical model** constrained by the encoder's "interaction hypothesis"

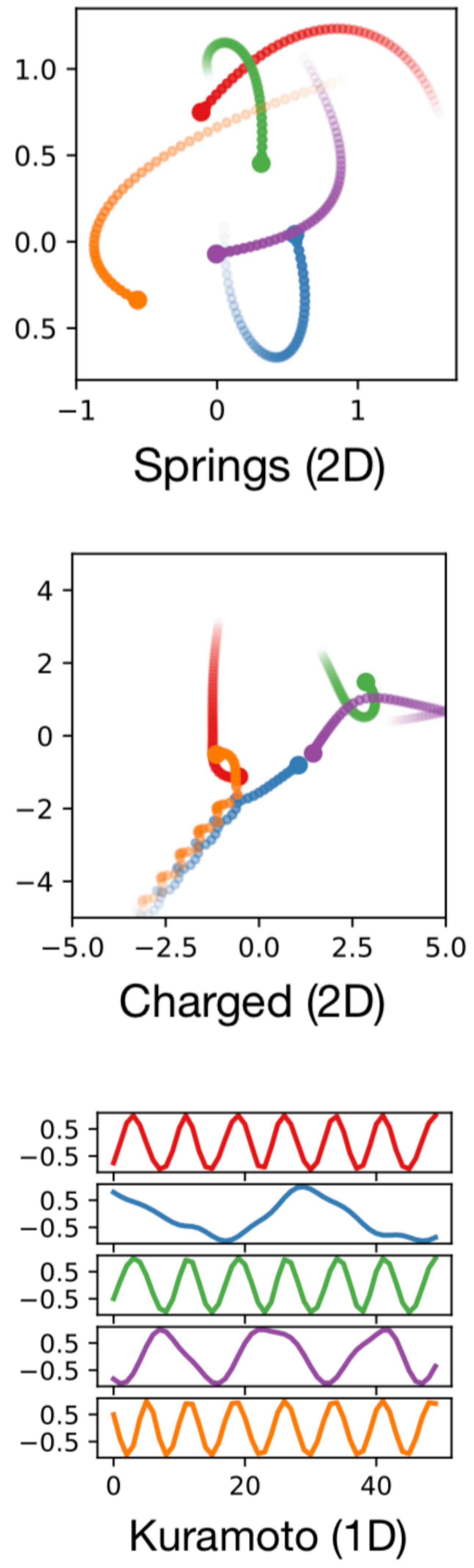
Trained jointly using **Gumbel softmax trick** as straight-through gradient estimator

Yang et al. (ICLR 2017), Maddison et al. (ICLR 2017)

Learning latent interaction graphs

Table 1. Accuracy (in %) of unsupervised interaction recovery.

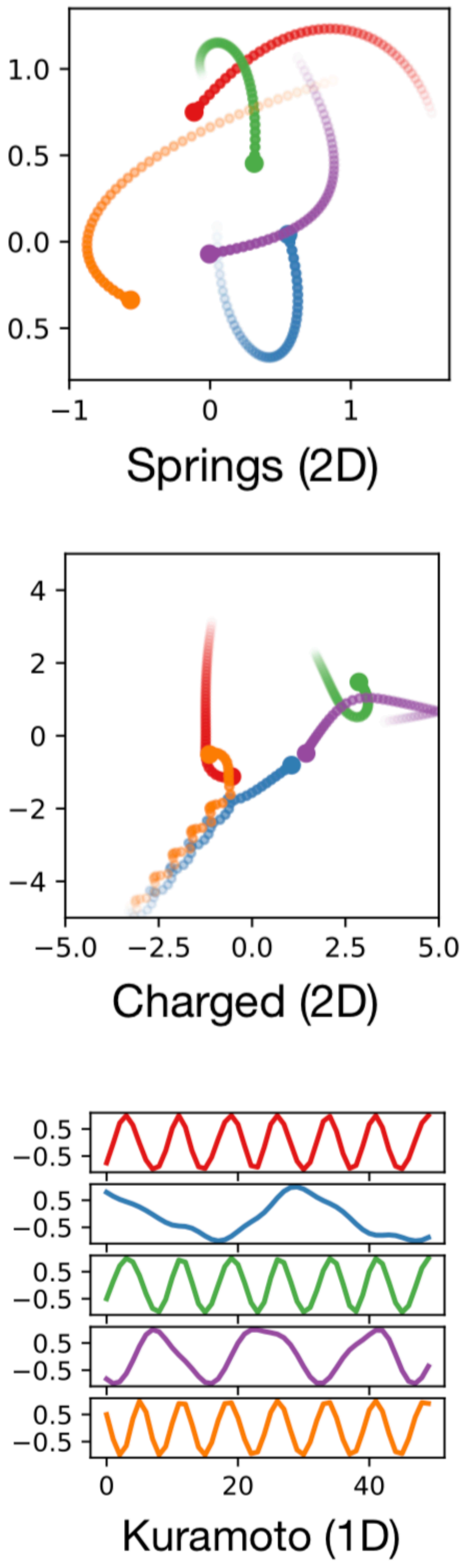
Model	Springs	Charged	Kuramoto
	5 objects		
Corr. (path)	52.4	55.8	62.8
Corr. (LSTM)	55.0	61.8	55.9
NRI (sim.)	**99.9**	59.1	–
NRI (learned)	**99.8**	**82.6**	**96.0**
Supervised	99.9	96.3	99.8
	10 objects		
Corr. (path)	50.4	51.4	59.3
Corr. (LSTM)	51.0	52.9	53.9
NRI (sim.)	**98.2**	53.9	–
NRI (learned)	**98.4**	**71.3**	**76.2**
Supervised	98.7	94.9	96.9



Learning latent interaction graphs

Table 1. Accuracy (in %) of unsupervised interaction recovery.

Model	Springs	Charged	Kuramoto
	5 objects		
Corr. (path)	52.4	55.8	62.8
Corr. (LSTM)	55.0	61.8	55.9
NRI (sim.)	**99.9**	59.1	–
NRI (learned)	**99.8**	**82.6**	**96.0**
Supervised	99.9	96.3	99.8
	10 objects		
Corr. (path)	50.4	51.4	59.3
Corr. (LSTM)	51.0	52.9	53.9
NRI (sim.)	**98.2**	53.9	–
NRI (learned)	**98.4**	**71.3**	**76.2**
Supervised	98.7	94.9	96.9

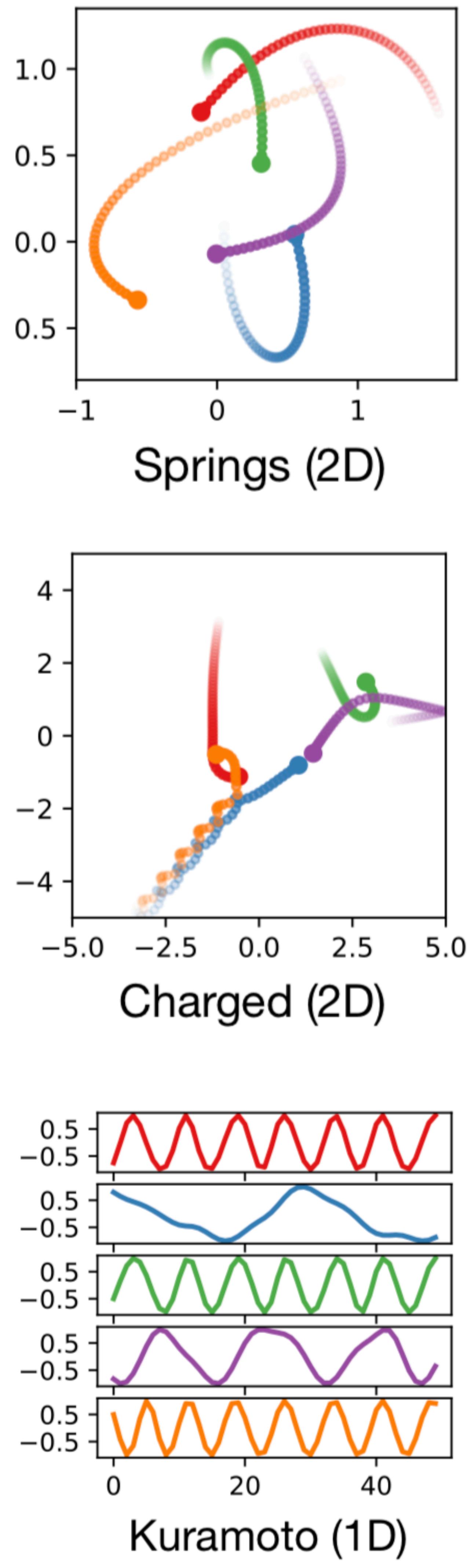


NRI can learn to discover ground-truth relations with very high accuracy!

Learning latent interaction graphs

Table 1. Accuracy (in %) of unsupervised interaction recovery.

Model	**Springs**	**Charged**	**Kuramoto**
	5 objects		
Corr. (path)	52.4	55.8	62.8
Corr. (LSTM)	55.0	61.8	55.9
NRI (sim.)	**99.9**	59.1	–
NRI (learned)	**99.8**	**82.6**	**96.0**
Supervised	99.9	96.3	99.8
	10 objects		
Corr. (path)	50.4	51.4	59.3
Corr. (LSTM)	51.0	52.9	53.9
NRI (sim.)	**98.2**	53.9	–
NRI (learned)	**98.4**	**71.3**	**76.2**
Supervised	98.7	94.9	96.9

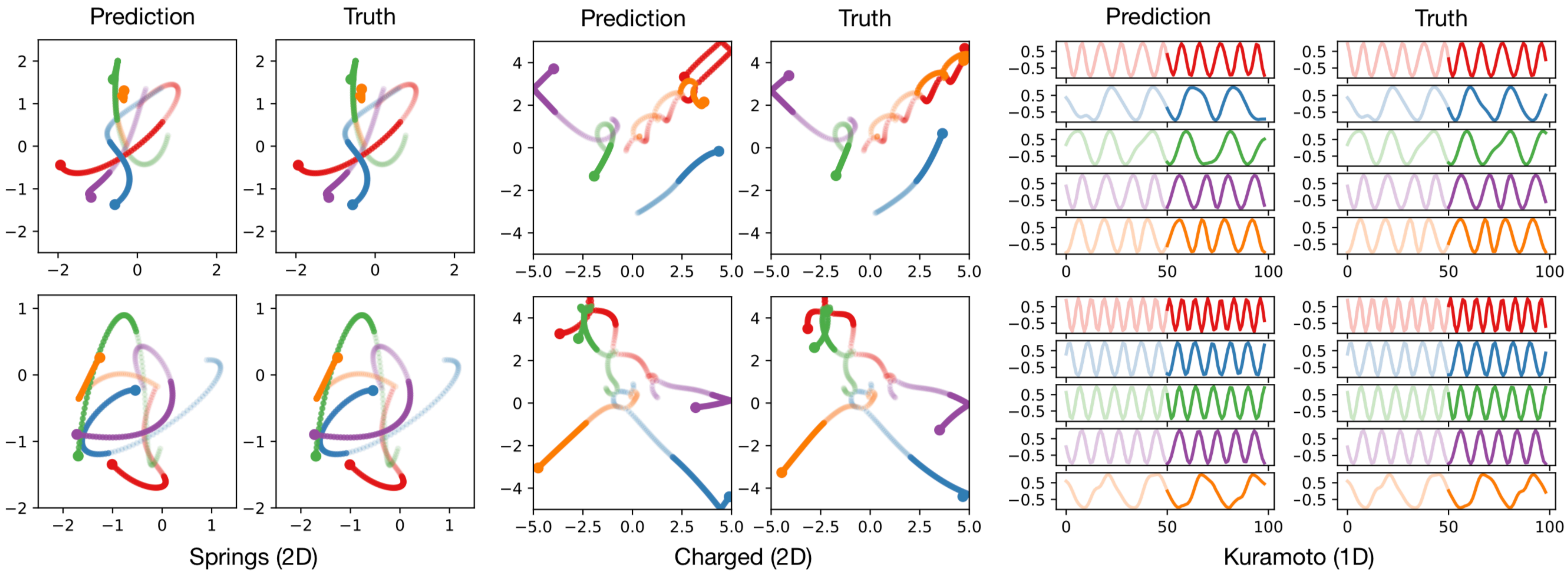


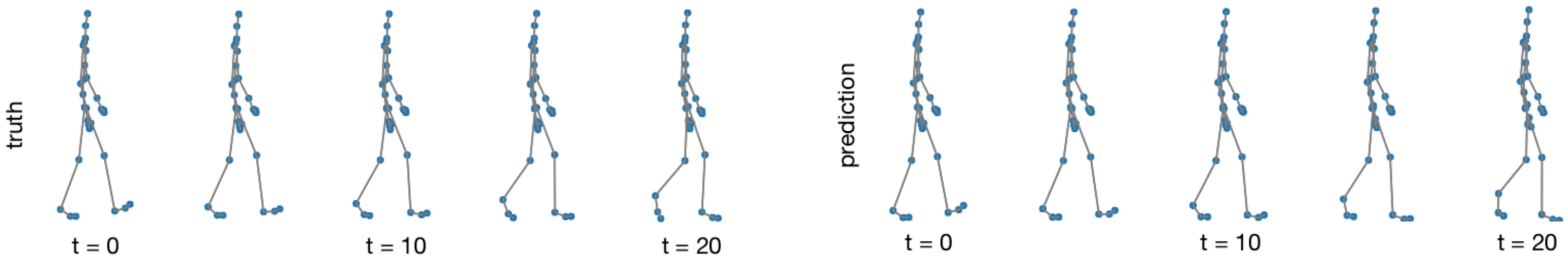
NRI can learn to discover ground-truth relations with very high accuracy!

Potential applications:

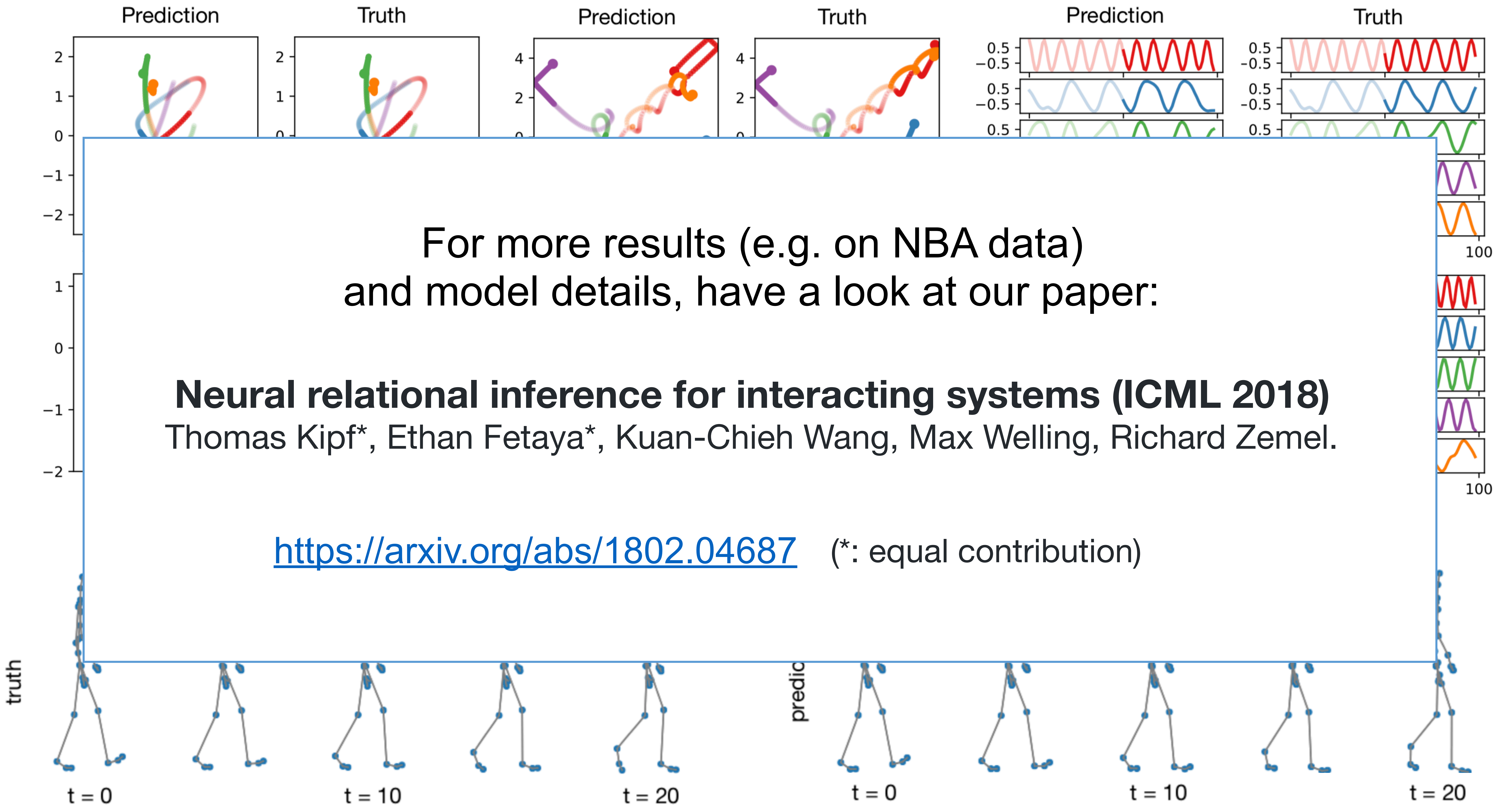
- Inference of causal relations
- Discovering protein interaction networks
- Program induction (with programs as graphs)

Qualitative results





Qualitative results



For more results (e.g. on NBA data)
and model details, have a look at our paper:

Neural relational inference for interacting systems (ICML 2018)
Thomas Kipf*, Ethan Fetaya*, Kuan-Chieh Wang, Max Welling, Richard Zemel.

https://arxiv.org/abs/1802.04687 (*: equal contribution)

Deep generative models of graphs

MolGAN: An implicit generative model for small molecular graphs

Nicola De Cao [1] Thomas Kipf [1]

Abstract

Deep generative models for graph-structured data offer a new angle on the problem of chemical synthesis: by optimizing differentiable models that directly generate molecular graphs, it is possible to side-step expensive search procedures in the discrete and vast space of chemical structures. We introduce MolGAN, an implicit, likelihood-free generative model for small molecular graphs that circumvents the need for expensive graph matching procedures or node ordering heuristics of previous likelihood-based methods. Our method adapts generative adversarial networks (GANs) to operate directly on graph-structured

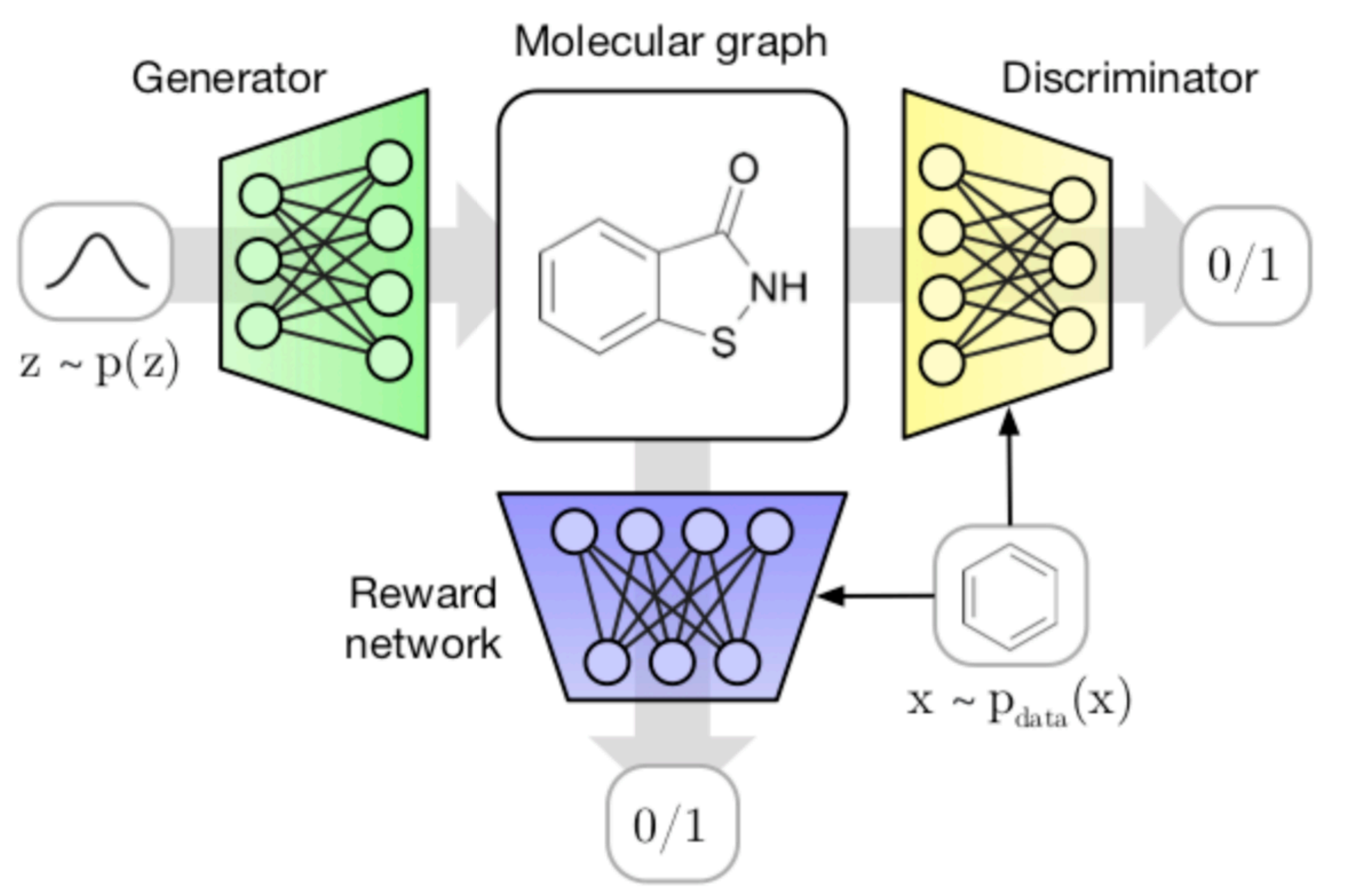


Figure 1. Schema of MolGAN. A vector z is sampled from a prior

With **Nicola De Cao**

(Under review at ICML TADGM Workshop)

arXiv version: **https://arxiv.org/abs/1805.11973**

Likelihood-based (deep) graph generation

Version 1: Generate graph (or predict new links) between known entities

Graph-based autoencoders:

- Encoder: GNN/GCN
- Decoder: Pairwise scoring function

$$p(A_{ij}) = f(\mathbf{z}_i, \mathbf{z}_j)$$

e.g. $$p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$$

Likelihood-based (deep) graph generation

Version 1: Generate graph (or predict new links) between known entities

Graph-based autoencoders:

- Encoder: GNN/GCN
- Decoder: Pairwise scoring function

$$p(A_{ij}) = f(\mathbf{z}_i, \mathbf{z}_j)$$

e.g. $$p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$$

Likelihood-based:

- we have some ground truth graphs
- define likelihood as how well a generated graph matches a ground truth graph

Likelihood-based (deep) graph generation

Version 1: Generate graph (or predict new links) between known entities

Graph-based autoencoders:

- Encoder: GNN/GCN
- Decoder: Pairwise scoring function

$$p(A_{ij}) = f(\mathbf{z}_i, \mathbf{z}_j)$$

e.g. $$p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$$

Likelihood-based:

- we have some ground truth graphs
- define likelihood as how well a generated graph matches a ground truth graph

(Incomplete) History:

(Variational) Graph Auto-Encoders
Kipf & Welling
(NIPS BDL 2016)

→ Graphite
Grover et al.
(NIPS BDL 2017)

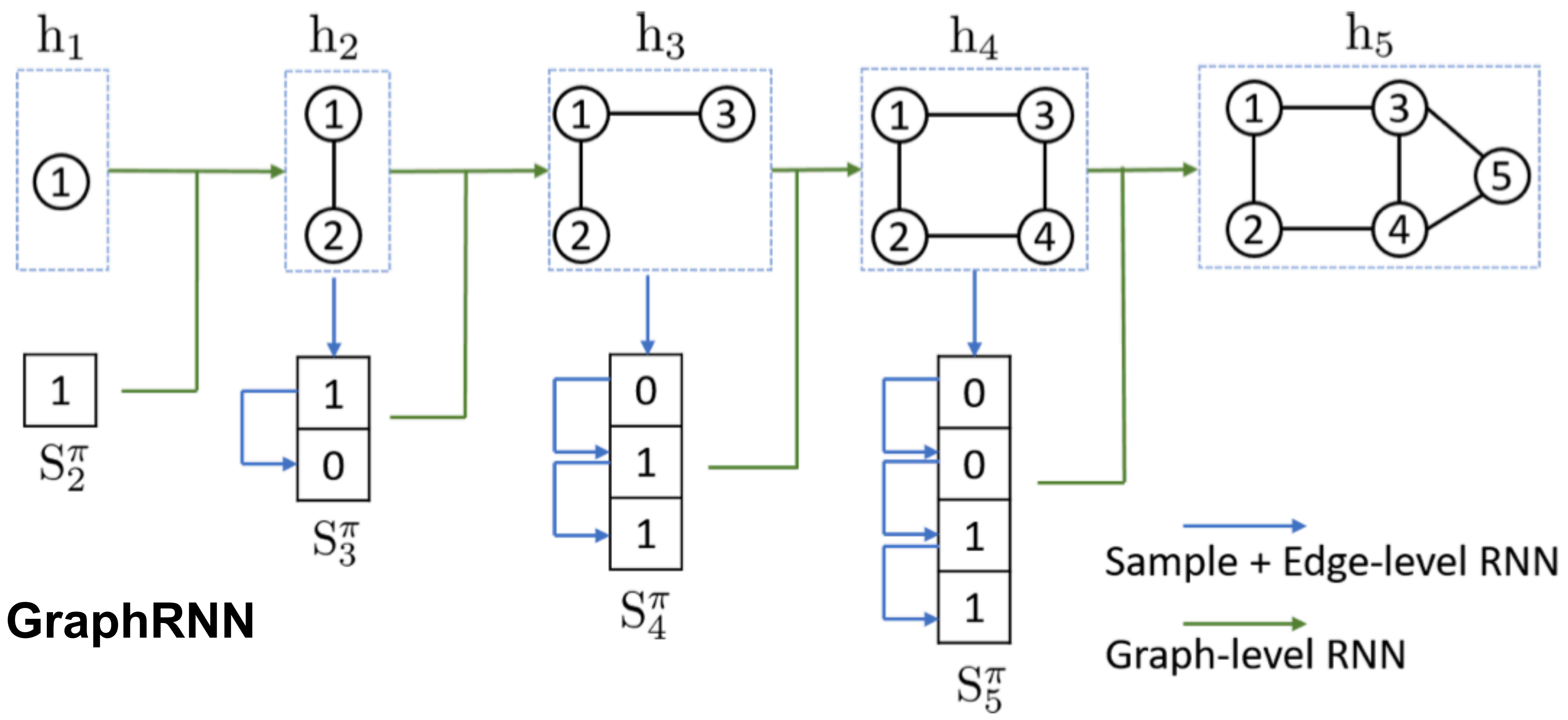
→ Graph2Gauss
Bojchevski & Günnemann
(ICLR 2018)

→ Hyperspherical VAEs
Davidson et al.
(UAI 2018)

Likelihood-based (deep) graph generation

Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:

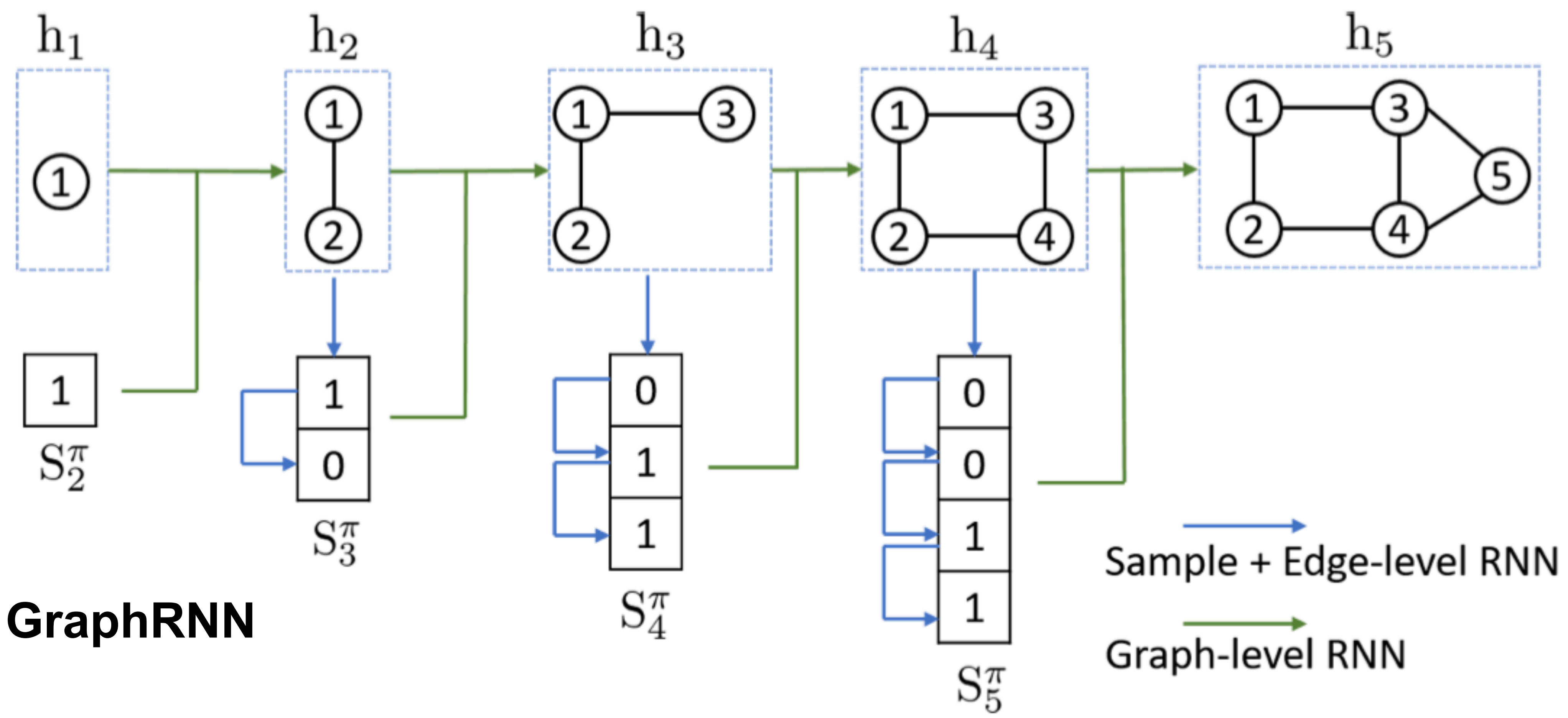


GraphRNN

Likelihood-based (deep) graph generation

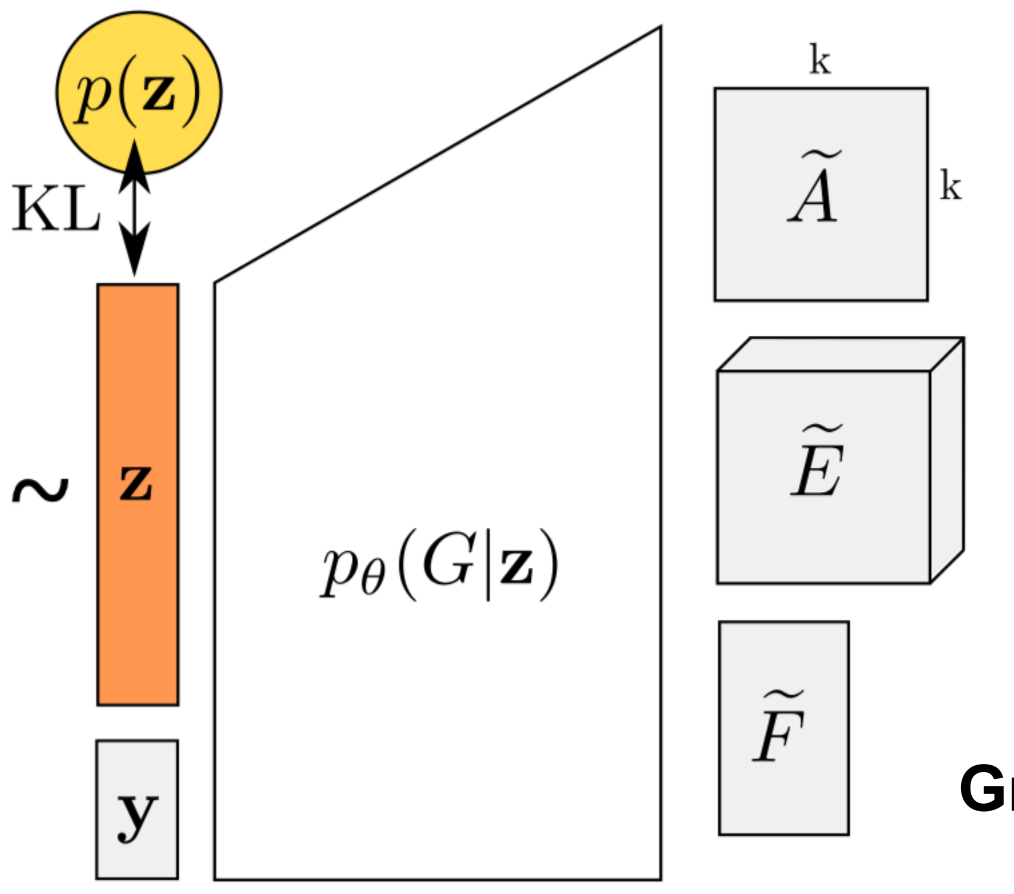
Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:



GraphRNN

Or in a single step:

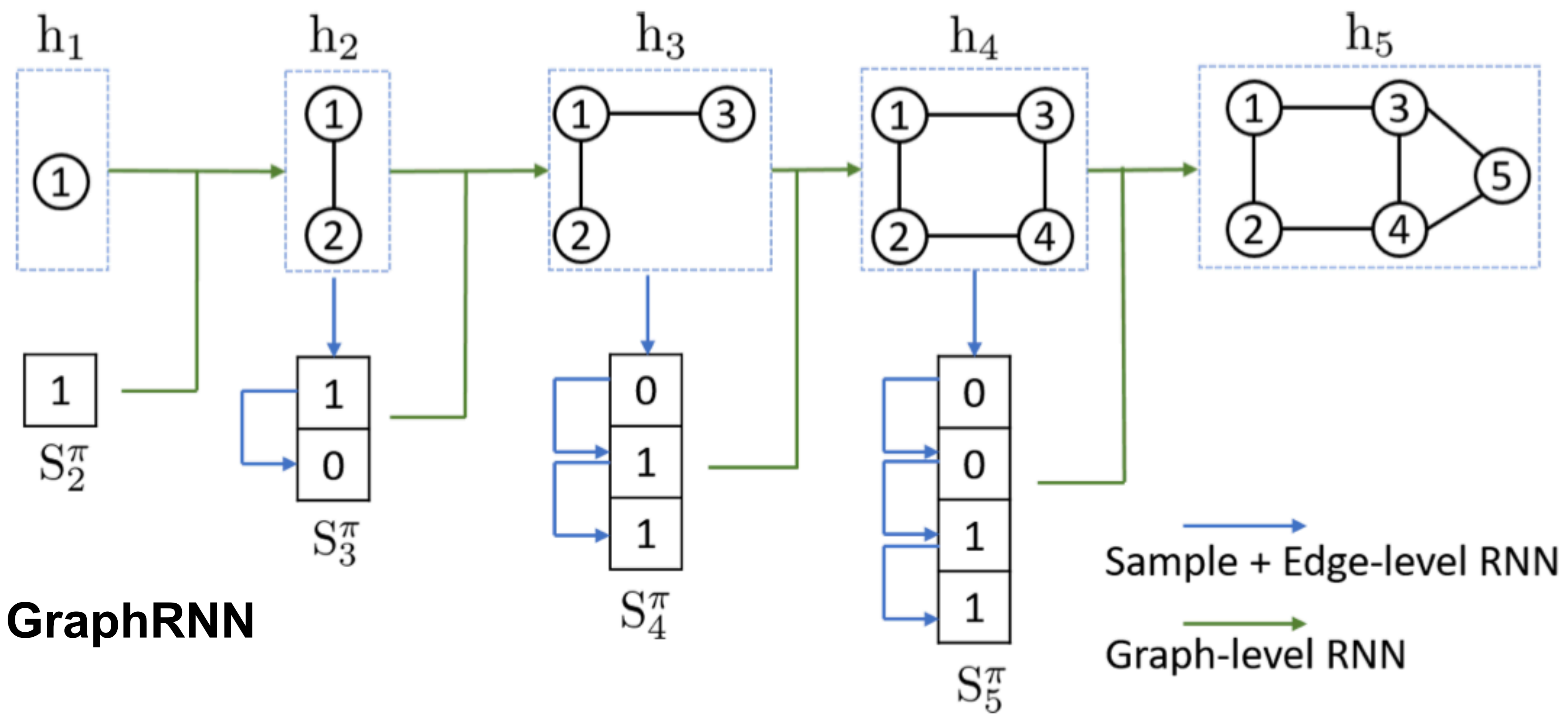


GraphVAE

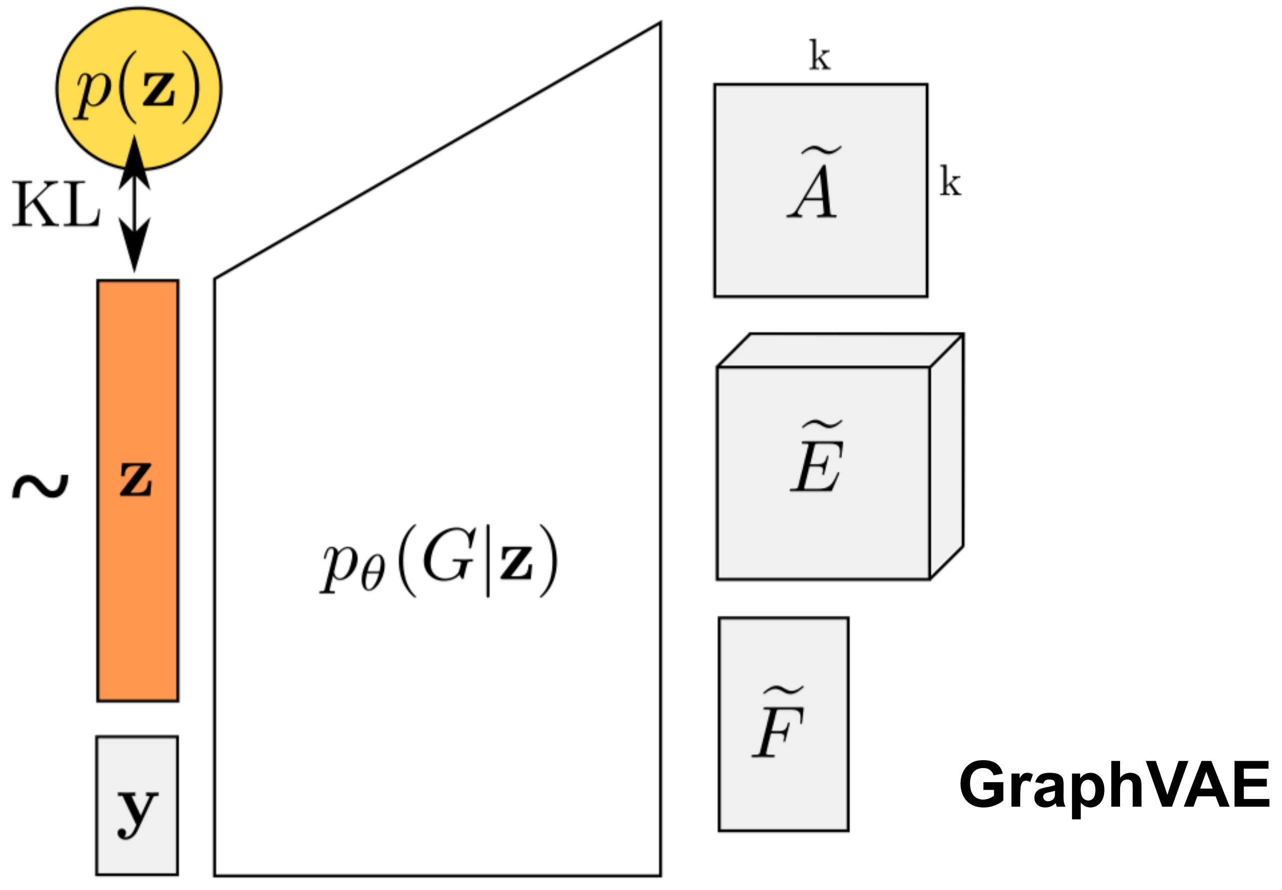
Likelihood-based (deep) graph generation

Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:



Or in a single step:



Learning Graphical State Transitions
Johnson
(ICLR 2017)

Deep Generative Models of Graphs
Li et al.
(arXiv 2018)

GraphVAE
Simonovsky et al.
(arXiv 2018)

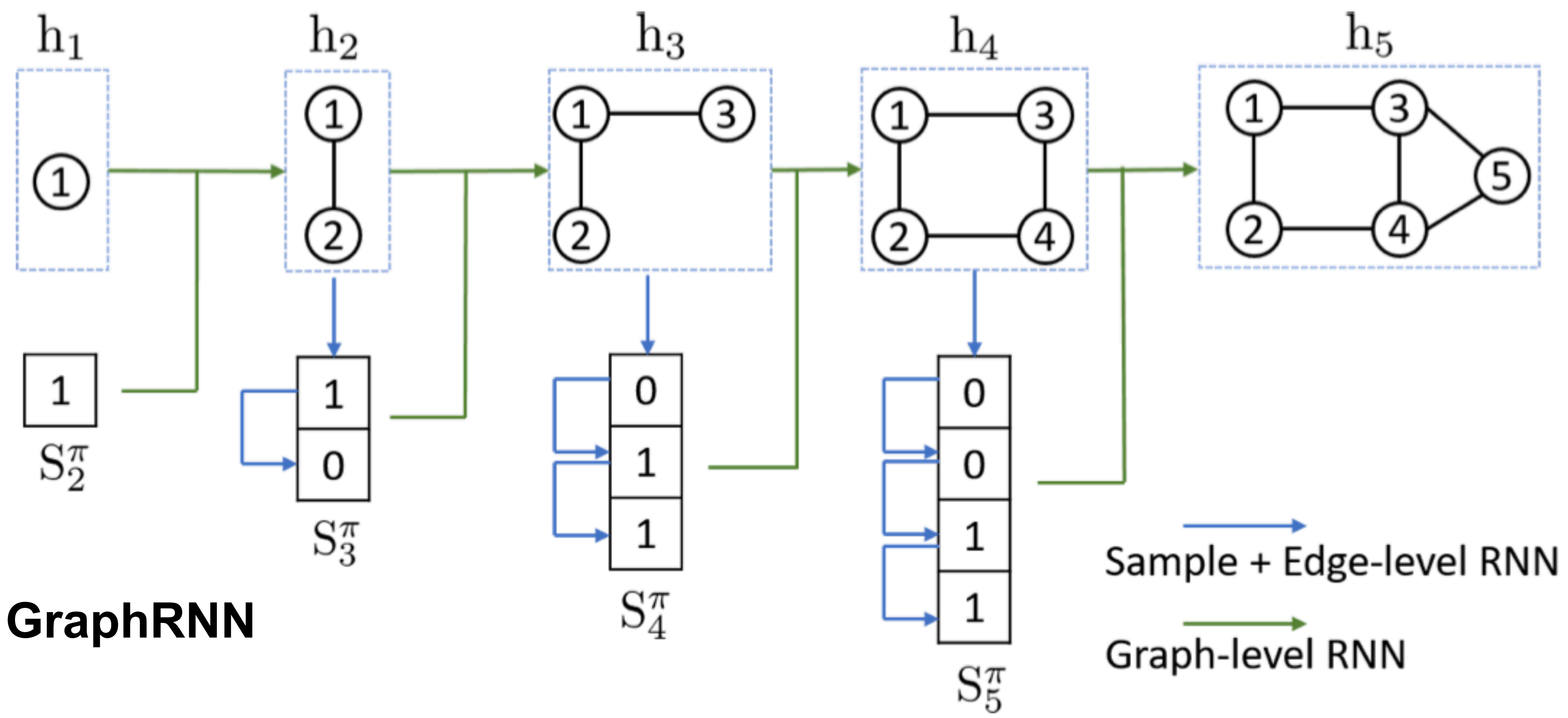
GraphRNN
You et al.
(ICML 2018)

Likelihood-based (deep) graph generation

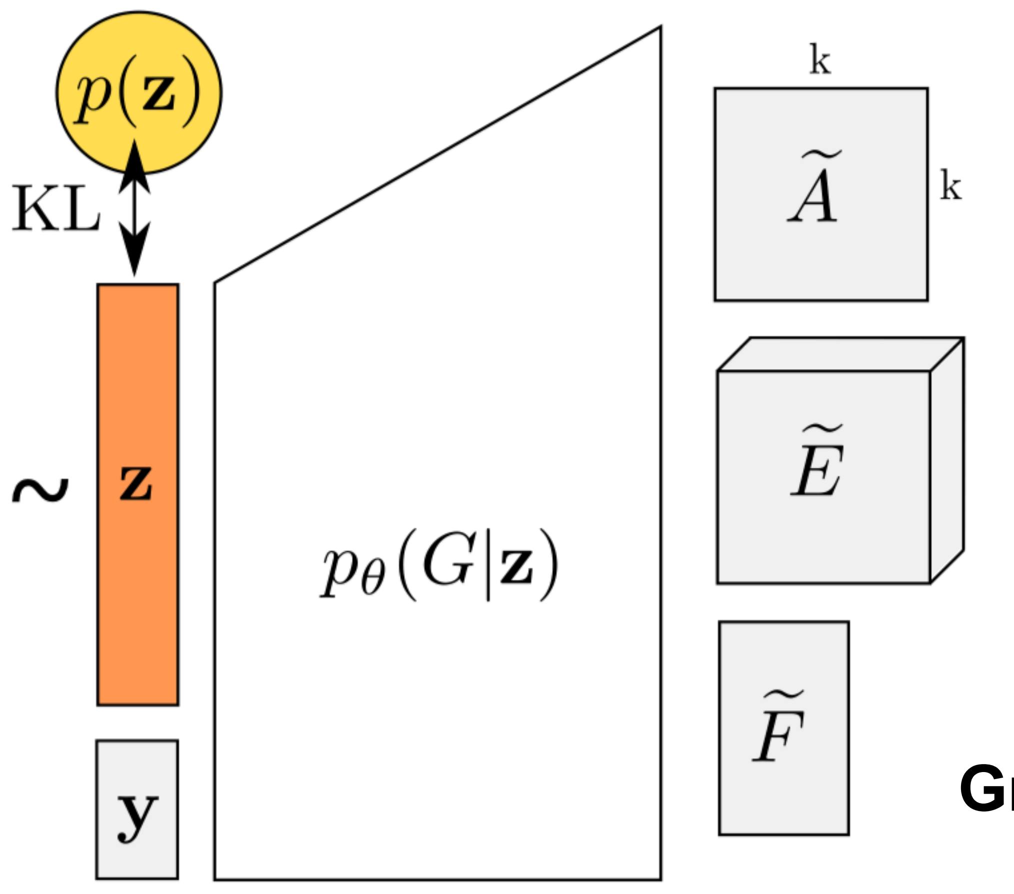
Version 2: Generate graphs from scratch (single embedding vector)

Or in a single step:

Sequentially:



GraphRNN



GraphVAE

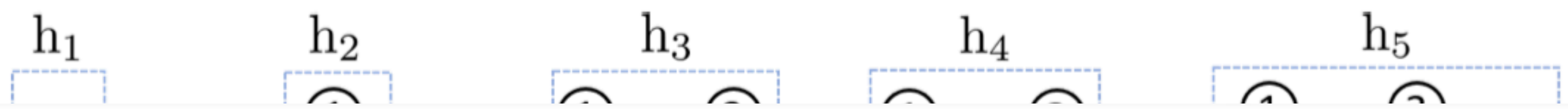
Learning Graphical State Transitions Johnson (ICLR 2017)	Deep Generative Models of Graphs Li et al. (arXiv 2018)	GraphVAE Simonovsky et al. (arXiv 2018)	GraphRNN You et al. (ICML 2018)

Both 1-2 times rejected… Seems to be hard to get these ideas published!

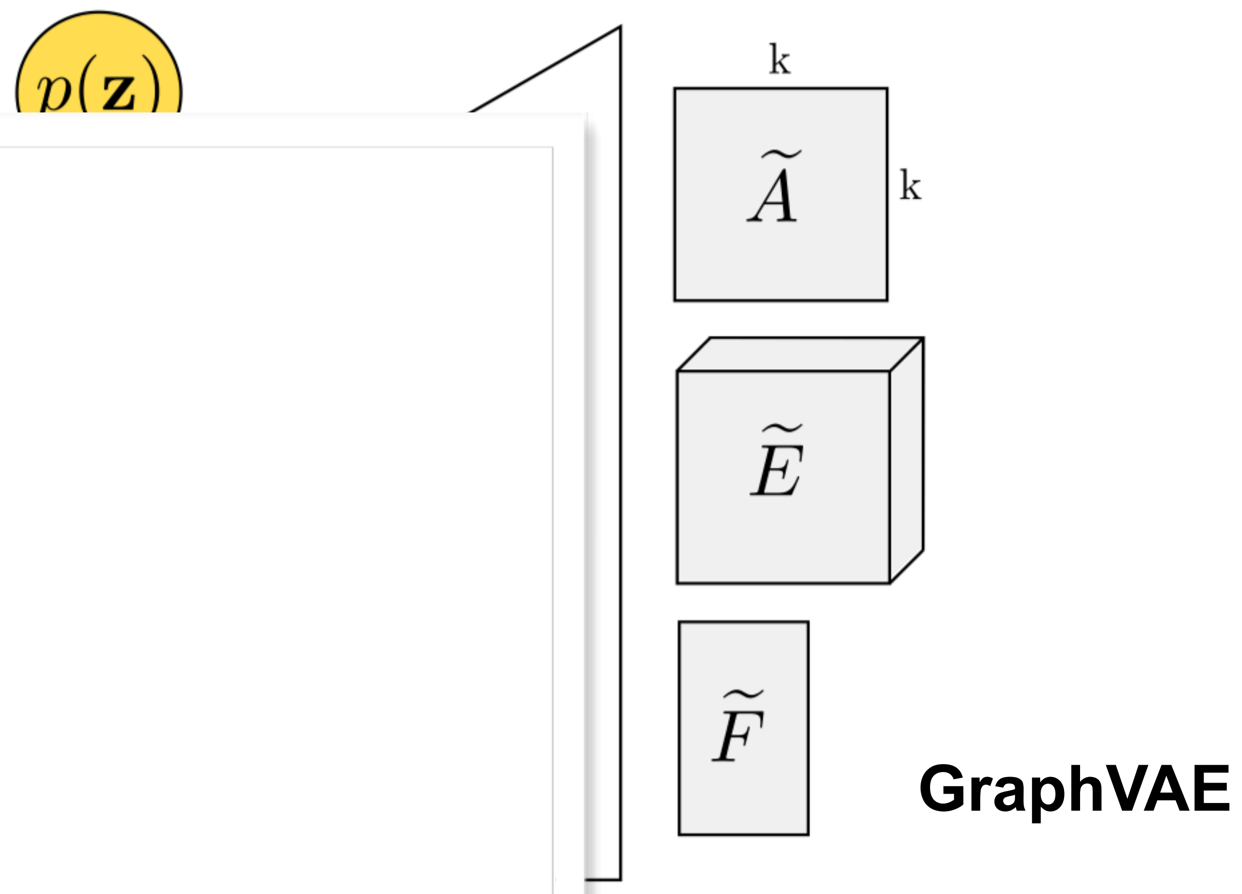
Likelihood-based (deep) graph generation

Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:



Or in a single step:



Problem:

Need to define some node ordering or perform (expensive) graph matching

(evaluating all possible permutations is too expensive)

Learning Graphical State Transitions Johnson (ICLR 2017)	Deep Generative Models of Graphs Li et al. (arXiv 2018)	GraphVAE Simonovsky et al. (arXiv 2018)	GraphRNN You et al. (ICML 2018)

Both 1-2 times rejected… Seems to be hard to get these ideas published!

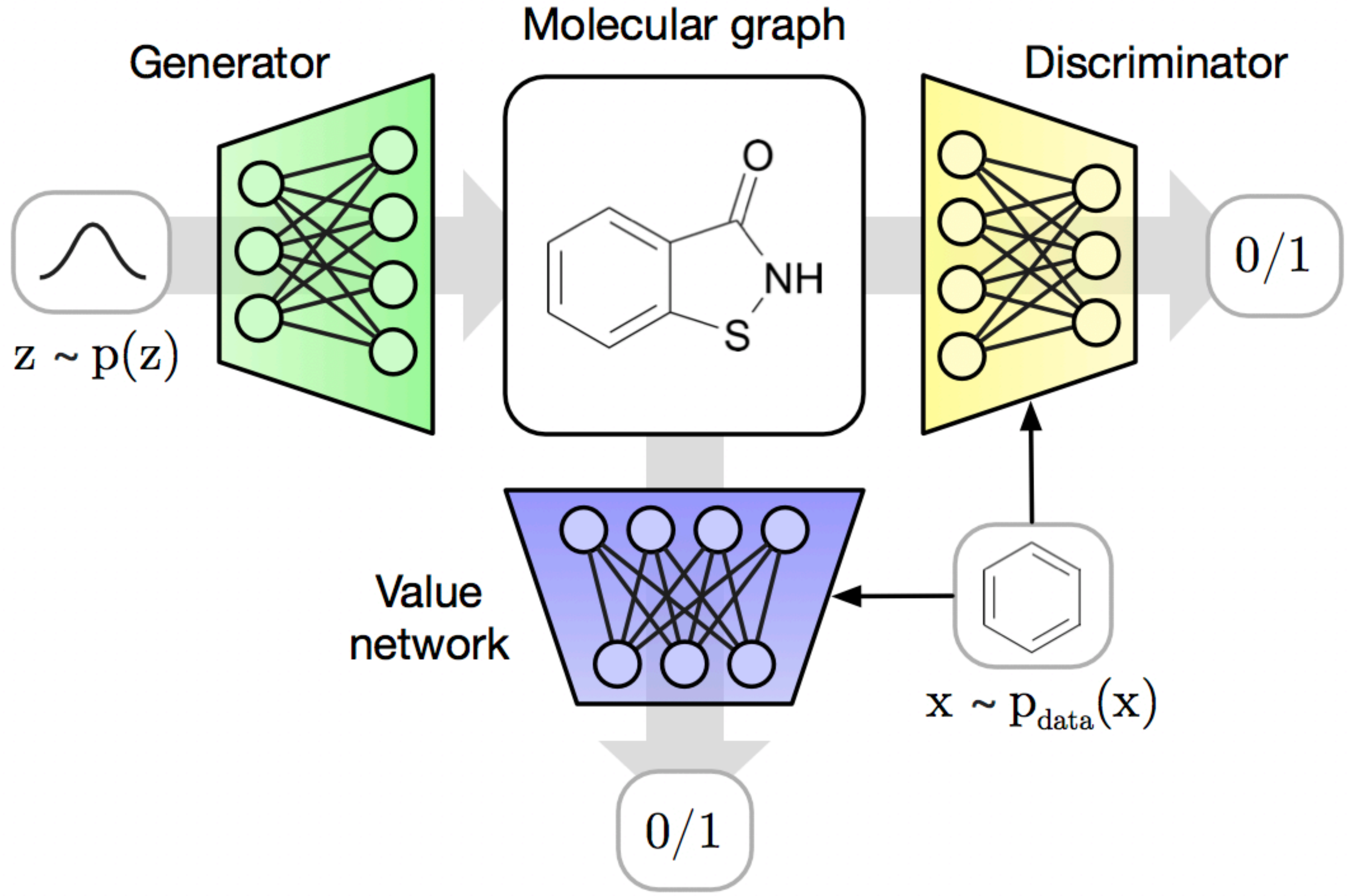
MolGAN model

Implicit models offer promising alternative:
No graph matching / fixed ordering needed!

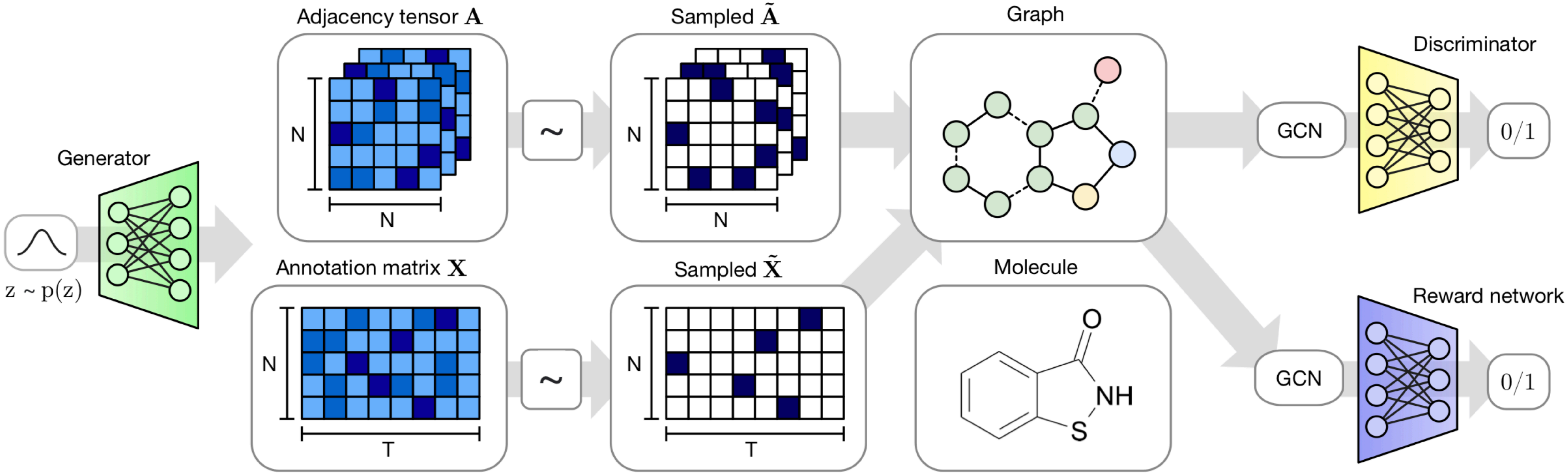
In practice: Use **GANs** and/or **RL**

MolGAN:

- Couple graph generator to a **discriminator** and a **reward network**
- Train discriminator via GAN objective
- Train reward net via RL objective
- Generator is trained jointly



MolGAN model architecture

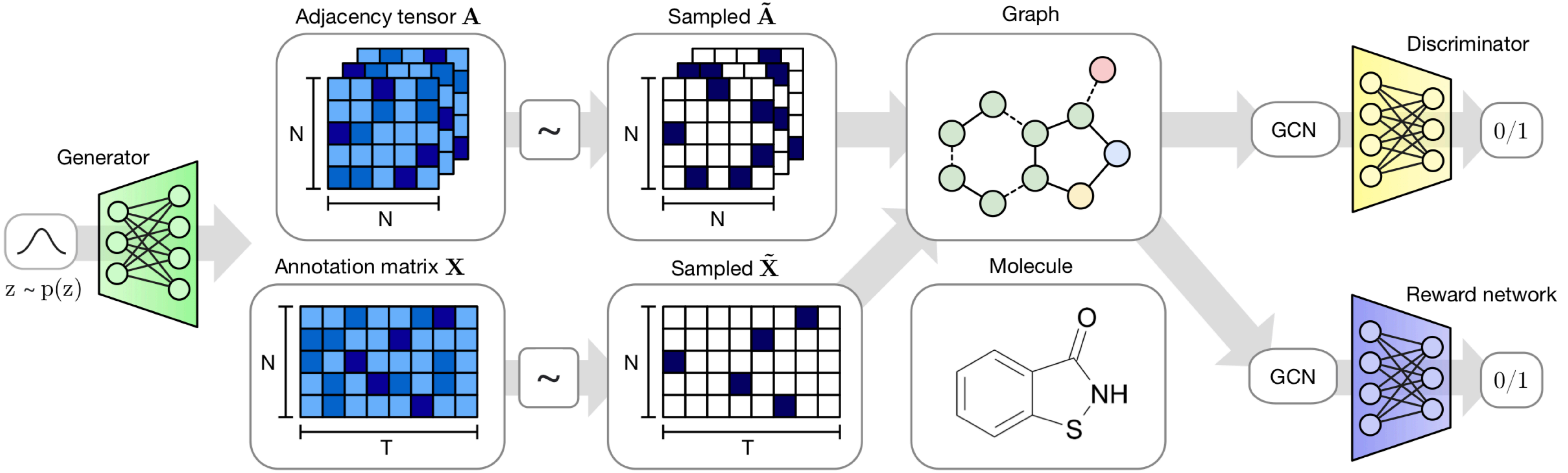


Generator: MLP to predict graph at once

Discriminator / reward net: GNN/GCN (with support for multiple edge types)

$$\begin{cases} \boldsymbol{h}_i'^{(\ell+1)} = f_s(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{x}_i) + \sum_{j=1}^{N} \sum_{y=1}^{Y} \frac{\boldsymbol{A}_{ijy}}{|\mathcal{N}_i|} f_y(\boldsymbol{h}_j^{(\ell)}, \boldsymbol{x}_i) \\ \boldsymbol{h}_i^{(\ell+1)} = \tanh(\boldsymbol{h}_i'^{(\ell+1)}) , \end{cases}$$

MolGAN model architecture



Generator: MLP to predict graph at once

Discriminator / reward net: GNN/GCN (with support for multiple edge types)

$$\begin{cases} \boldsymbol{h}_i'^{(\ell+1)} = f_s(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{x}_i) + \sum_{j=1}^{N} \sum_{y=1}^{Y} \frac{\boldsymbol{A}_{ijy}}{|\mathcal{N}_i|} f_y(\boldsymbol{h}_j^{(\ell)}, \boldsymbol{x}_i) \\ \boldsymbol{h}_i^{(\ell+1)} = \tanh(\boldsymbol{h}_i'^{(\ell+1)}) , \end{cases}$$

+ gated global pooling

Li et al., (ICLR 2016)

$$\boldsymbol{h}_{\mathcal{G}}' = \sum_{v \in \mathcal{V}} \sigma(i(\boldsymbol{h}_v^{(L)}, \boldsymbol{x}_v)) \odot \tanh(j(\boldsymbol{h}_v^{(L)}, \boldsymbol{x}_v))$$

Chemical synthesis results

ORGAN: GAN-based sequence generation model, Guimaraes et al. (2017)

Objective	Algorithm	Valid (%)	Unique (%)	Time (h)	Diversity	Druglikeliness	Synthesizability	Solubility
Druglikeliness	ORGAN	88.2	-	-	0.55	0.52	0.32	0.35
	OR(W)GAN	85.0	8.2*	10.06*	0.95	0.60	0.54	0.47
	Naive RL	97.1	-	-	0.80	0.57	0.53	0.50
	MolGAN	99.9	2.0	1.66	0.95	0.61	0.68	0.52
	MolGAN (QM9)	**100.0**	2.2	4.12	**0.97**	**0.62**	0.59	0.53
Synthesizability	ORGAN	96.5	-	-	0.92	0.51	0.83	0.45
	OR(W)GAN	97.6	30.7*	9.60*	**1.00**	0.20	0.75	0.84
	Naive RL	97.7	-	-	0.96	0.52	0.83	0.46
	MolGAN	99.4	2.1	1.04	0.75	0.52	0.90	0.67
	MolGAN (QM9)	**100.0**	2.1	2.49	0.95	0.53	**0.95**	0.68
Solubility	ORGAN	94.7	54.3*	8.65*	0.76	0.50	0.63	0.55
	OR(W)GAN	94.1	20.8*	9.21*	0.90	0.42	0.66	0.54
	Naive RL	92.7	-	-	0.75	0.49	0.70	0.78
	MolGAN	**99.8**	2.3	0.58	0.97	0.45	0.42	0.86
	MolGAN (QM9)	**99.8**	2.0	1.62	**0.99**	0.44	0.22	**0.89**

Chemical synthesis results

ORGAN: GAN-based sequence generation model, Guimaraes et al. (2017)

Objective	Algorithm	Valid (%)	Unique (%)	Time (h)	Diversity	Druglikeliness	Synthesizability	Solubility
Druglikeliness	ORGAN	88.2	-	-	0.55	0.52	0.32	0.35
	OR(W)GAN	85.0	8.2*	10.06*	0.95	0.60	0.54	0.47
	Naive RL	97.1	-	-	0.80	0.57	0.53	0.50
	MolGAN	99.9	2.0	1.66	0.95	0.61	0.68	0.52
	MolGAN (QM9)	**100.0**	2.2	4.12	**0.97**	**0.62**	0.59	0.53
Synthesizability	ORGAN	96.5	-	-	0.92	0.51	0.83	0.45
	OR(W)GAN	97.6	30.7*	9.60*	**1.00**	0.20	0.75	0.84
	Naive RL	97.7	-	-	0.96	0.52	0.83	0.46
	MolGAN	99.4	2.1	1.04	0.75	0.52	0.90	0.67
	MolGAN (QM9)	**100.0**	2.1	2.49	0.95	0.53	**0.95**	0.68
Solubility	ORGAN	94.7	54.3*	8.65*	0.76	0.50	0.63	0.55
	OR(W)GAN	94.1	20.8*	9.21*	0.90	0.42	0.66	0.54
	Naive RL	92.7	-	-	0.75	0.49	0.70	0.78
	MolGAN	**99.8**	2.3	0.58	0.97	0.45	0.42	0.86
	MolGAN (QM9)	**99.8**	2.0	1.62	**0.99**	0.44	0.22	**0.89**

Significantly faster training/generation

Chemical synthesis results

ORGAN: GAN-based sequence generation model, Guimaraes et al. (2017)

Objective	Algorithm	Valid (%)	Unique (%)	Time (h)	Diversity	Druglikeliness	Synthesizability	Solubility
Druglikeliness	ORGAN	88.2	-	-	0.55	0.52	0.32	0.35
	OR(W)GAN	85.0	8.2*	10.06*	0.95	0.60	0.54	0.47
	Naive RL	97.1	-	-	0.80	0.57	0.53	0.50
	MolGAN	99.9	2.0	1.66	0.95	0.61	0.68	0.52
	MolGAN (QM9)	**100.0**	2.2	4.12	**0.97**	**0.62**	0.59	0.53
Synthesizability	ORGAN	96.5	-	-	0.92	0.51	0.83	0.45
	OR(W)GAN	97.6	30.7*	9.60*	**1.00**	0.20	0.75	0.84
	Naive RL	97.7	-	-	0.96	0.52	0.83	0.46
	MolGAN	99.4	2.1	1.04	0.75	0.52	0.90	0.67
	MolGAN (QM9)	**100.0**	2.1	2.49	0.95	0.53	**0.95**	0.68
Solubility	ORGAN	94.7	54.3*	8.65*	0.76	0.50	0.63	0.55
	OR(W)GAN	94.1	20.8*	9.21*	0.90	0.42	0.66	0.54
	Naive RL	92.7	-	-	0.75	0.49	0.70	0.78
	MolGAN	**99.8**	2.3	0.58	0.97	0.45	0.42	0.86
	MolGAN (QM9)	**99.8**	2.0	1.62	**0.99**	0.44	0.22	**0.89**

Significantly faster training/generation

Mode collapse is still an issue

Conclusions

- **Deep Learning on graphs works!**
- Exciting area; lots of new applications and extensions (hard to catch up):

Conclusions

- **Deep Learning on graphs works!**
- Exciting area; lots of new applications and extensions (hard to catch up):

Relational reasoning

[Santoro et al., NIPS 2017]

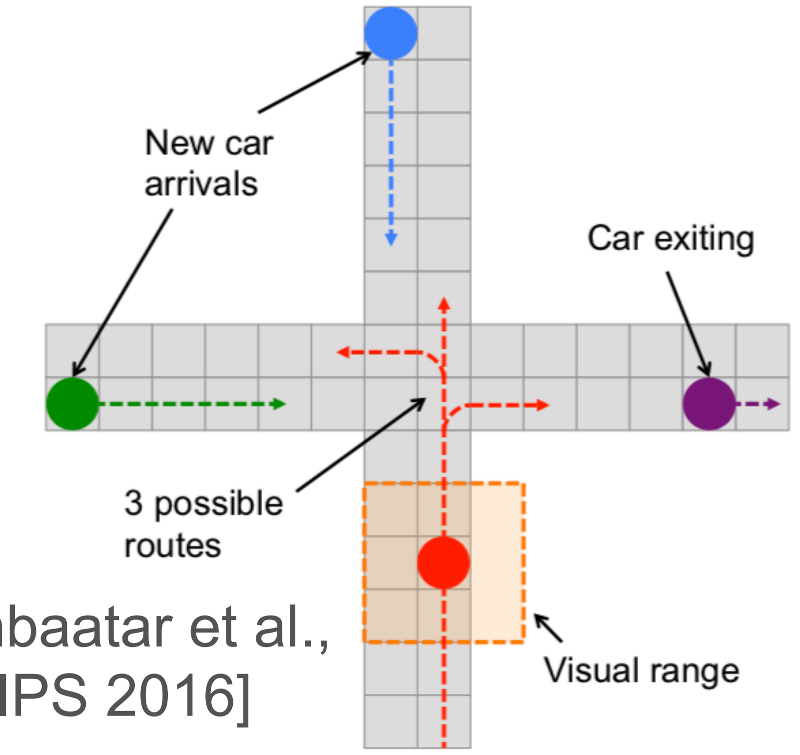
Conclusions

- **Deep Learning on graphs works!**
- Exciting area; lots of new applications and extensions (hard to catch up):

Relational reasoning

[Santoro et al., NIPS 2017]

Multi-Agent RL



[Sukhbaatar et al., NIPS 2016]

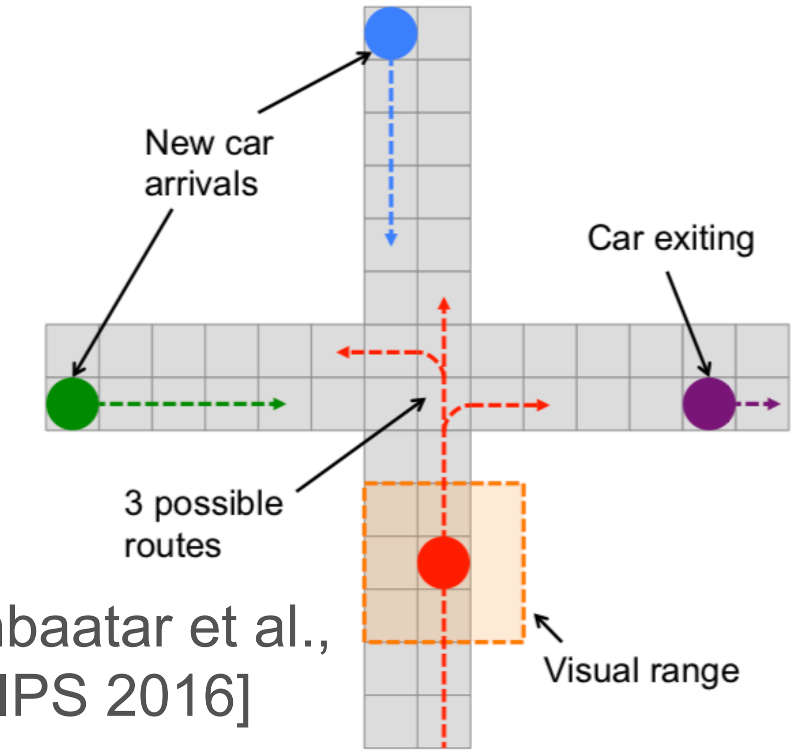
Conclusions

- **Deep Learning on graphs works!**
- Exciting area; lots of new applications and extensions (hard to catch up):

Relational reasoning

[Santoro et al., NIPS 2017]

Multi-Agent RL



[Sukhbaatar et al., NIPS 2016]

GCN for recommendation on 16 billion edge graph!

[Leskovec lab, Stanford]

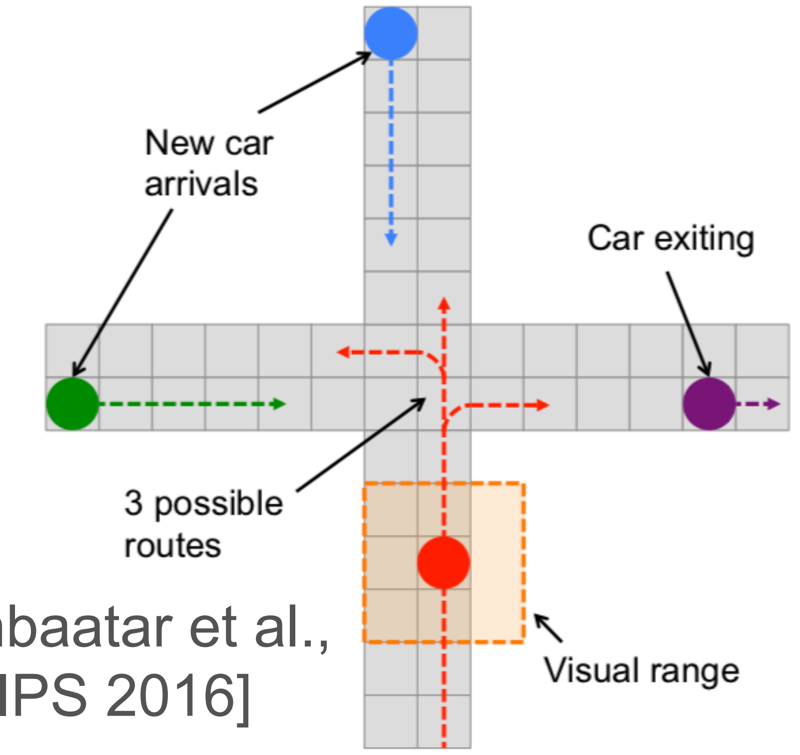
Conclusions

- **Deep Learning on graphs works!**
- Exciting area; lots of new applications and extensions (hard to catch up):

Relational reasoning

[Santoro et al., NIPS 2017]

Multi-Agent RL



[Sukhbaatar et al., NIPS 2016]

GCN for recommendation on 16 <u>billion</u> edge graph!

[Leskovec lab, Stanford]

Finding bugs in code

```
public ArraySegment<byte> ReadBytes(int length){
    int size = Math.Min(length, _len - _pos);
    var buffer = EnsureTempBuffer( length );
    var used = Read(buffer, 0, size);
```

[Allamanis et al., ICLR 2018]

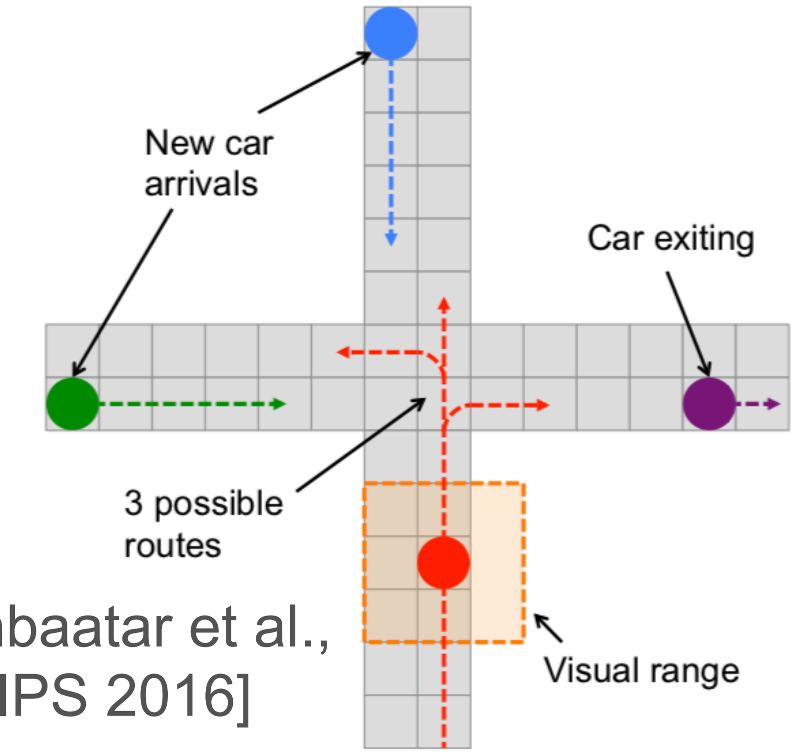
Conclusions

- **Deep Learning on graphs works!**
- Exciting area; lots of new applications and extensions (hard to catch up):

Relational reasoning

[Santoro et al., NIPS 2017]

Multi-Agent RL



[Sukhbaatar et al., NIPS 2016]

GCN for recommendation on 16 <u>billion</u> edge graph!

[Leskovec lab, Stanford]

Some open problems:

- Theory
- Scalable, stable generative models
- Learning on large, evolving data
- Multi-modal and cross-modal learning (e.g. sequence2graph etc.)

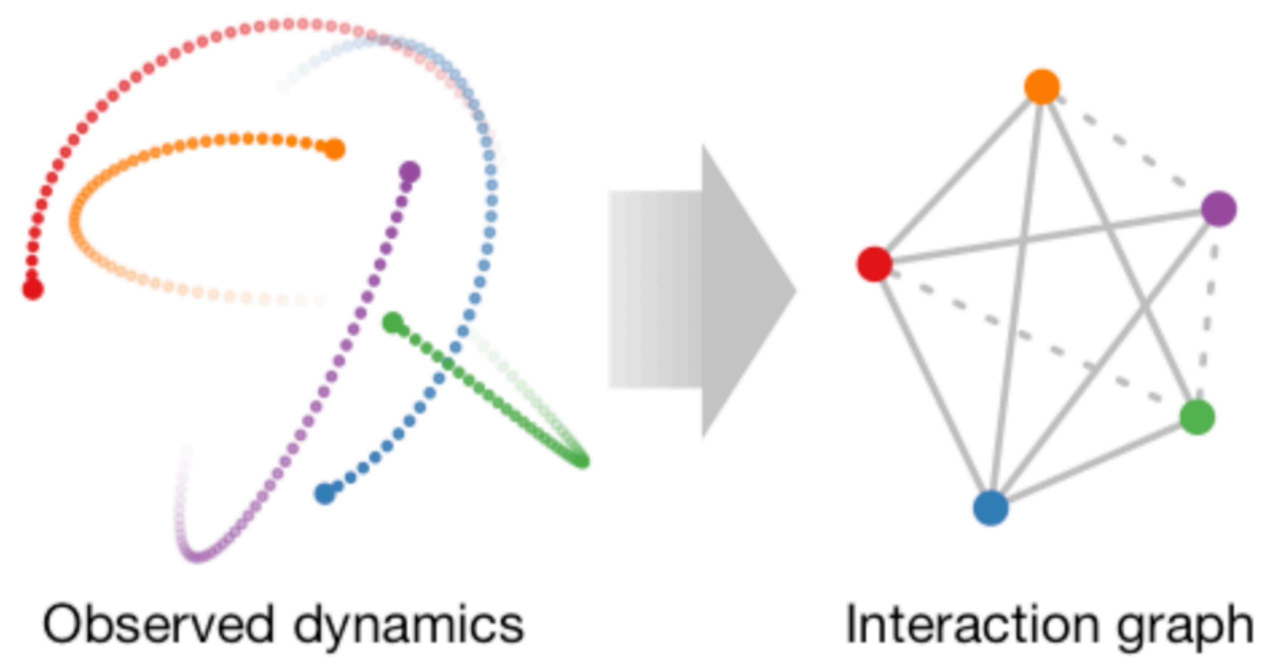
Finding bugs in code

```
public ArraySegment<byte> ReadBytes(int length){
    int size = Math.Min(length, _len - _pos);
    var buffer = EnsureTempBuffer( length );
    var used = Read(buffer, 0, size);
```

[Allamanis et al., ICLR 2018]

Further reading

Have a look at our ICML paper for an overview of recent work in the field:



Neural relational inference for interacting systems (ICML 2018)
Thomas Kipf*, Ethan Fetaya*, Kuan-Chieh Wang, Max Welling, Richard Zemel.

https://arxiv.org/abs/1802.04687 (*: equal contribution)

Code on Github:
http://github.com/ethanfetaya/nri

Other material:

Blog post on Graph Convolutional Networks:
http://tkipf.github.io/graph-convolutional-networks

GCN code on Github:
http://github.com/tkipf/gcn

Get in touch:
- E-Mail: t.n.kipf@uva.nl
- Twitter: @thomaskipf
- Web: tkipf.github.io

Project supported by SAP