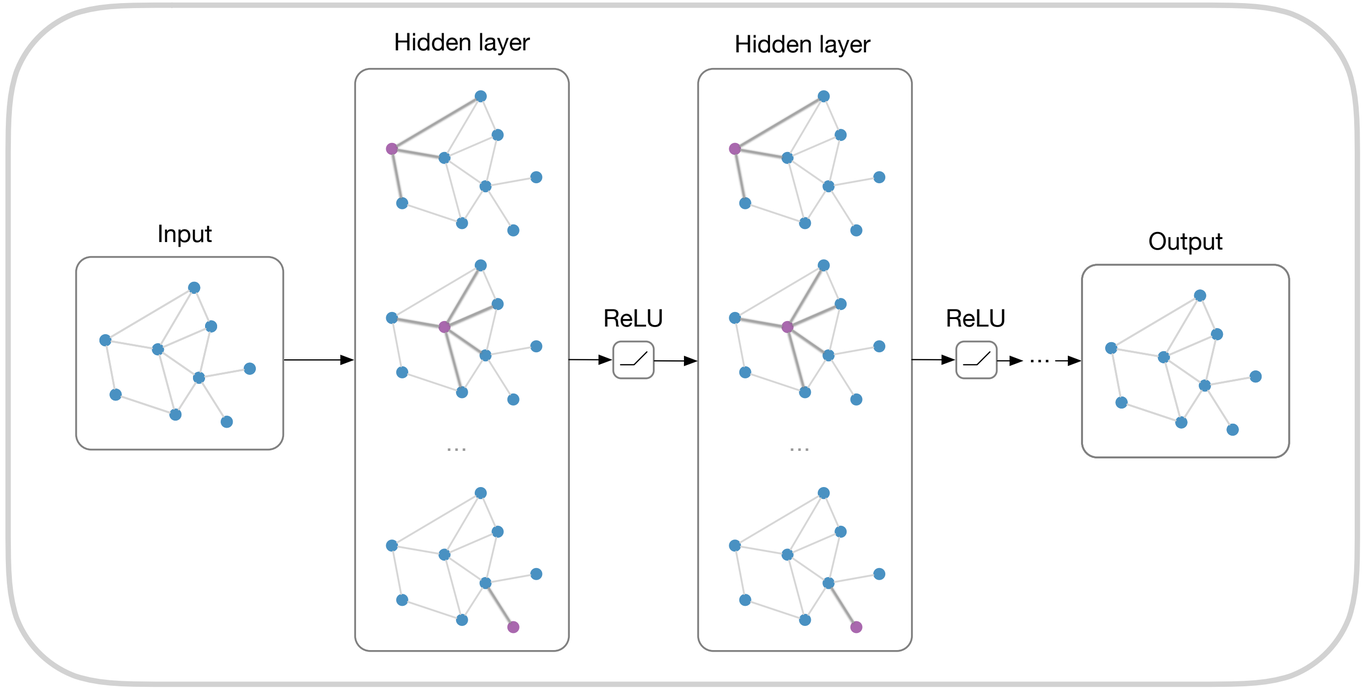
Unsupervised Learning with Graph Neural Networks



Thomas Kipf, 22 May 2019

UNIVERSITEIT VAN AMSTERDAM

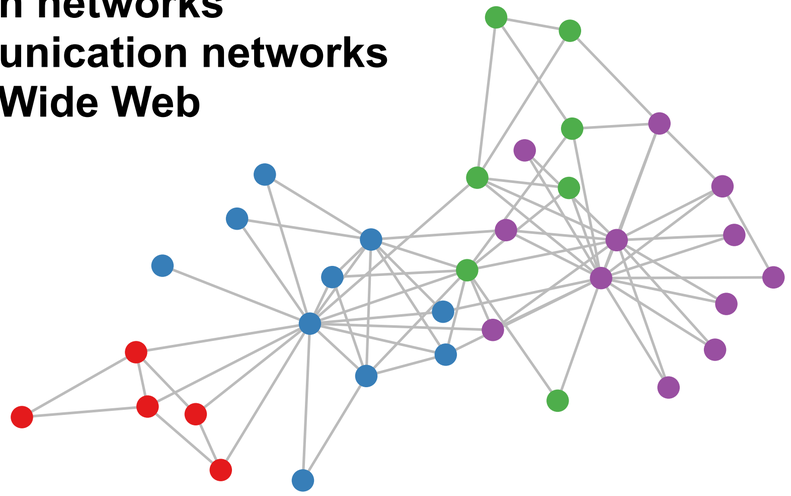
In collaboration with Ethan Fetaya, Daniel Daza, Rianne van den Berg, Michael Schlichtkrull, Ivan Titov, Max Welling, Richard Zemel and others

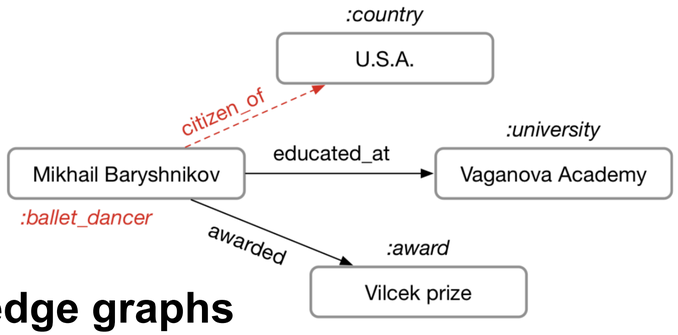
AMLAB Amsterdam Machine Learning Lab

Implicit structure: objects & relations as useful abstraction

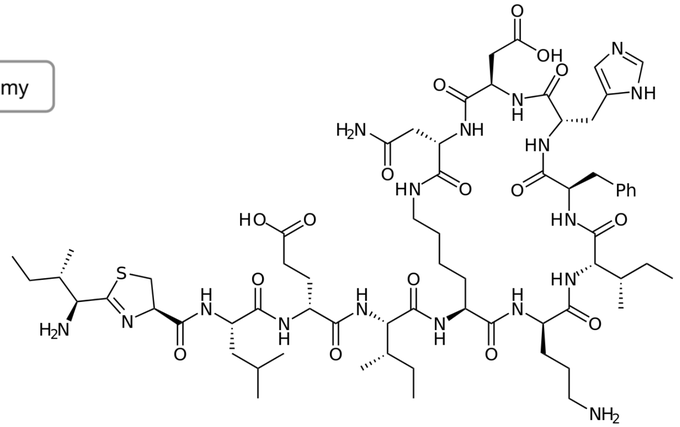
Explicitly graph-structured data

Social networks
Citation networks
Communication networks
World Wide Web

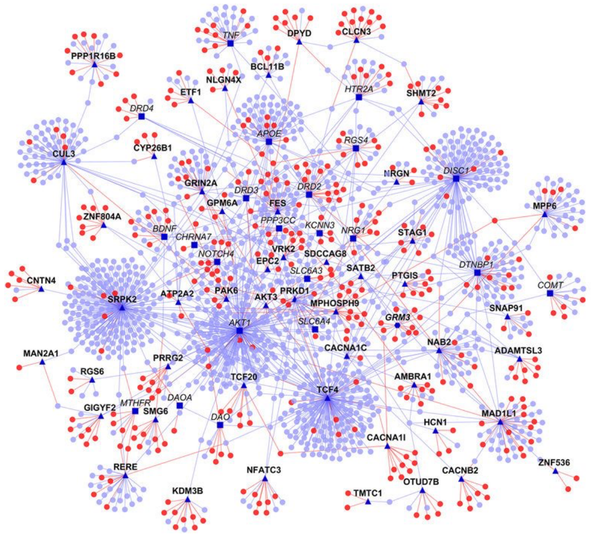




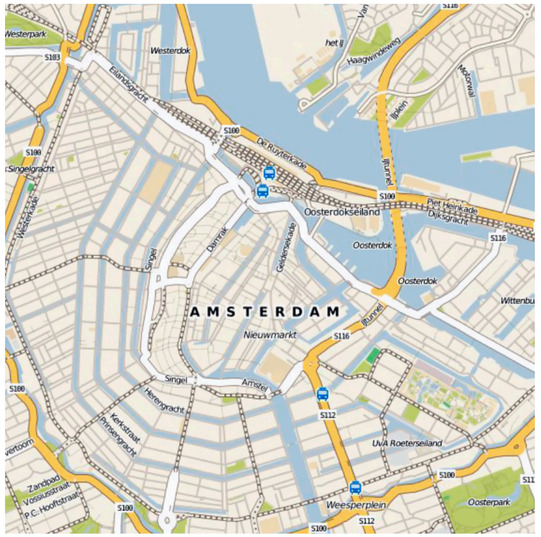
Knowledge graphs



Molecules



Protein interaction networks



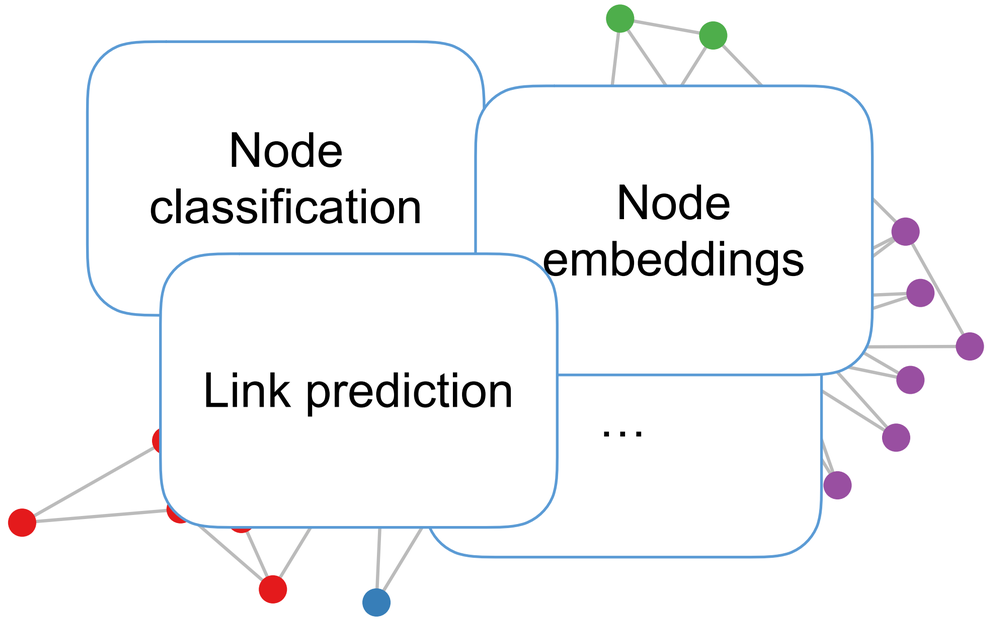
Road maps

Challenging for standard deep neural net architectures (CNNs / RNNs)

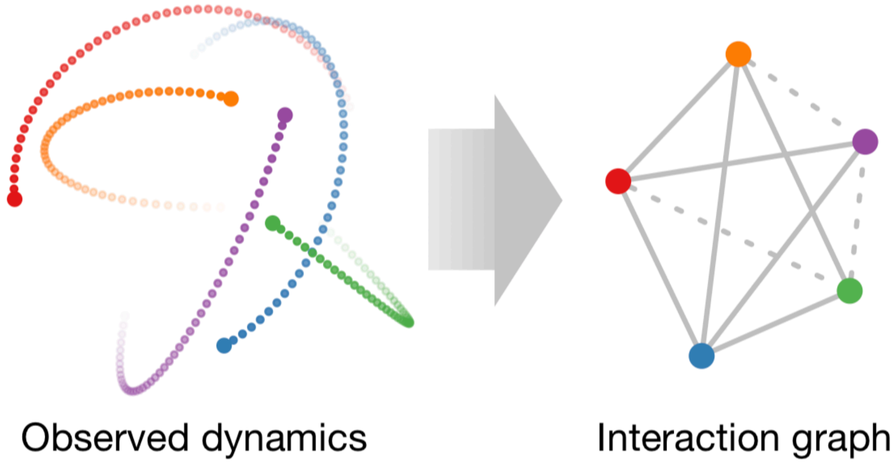
Talk overview

1) Graph neural networks (GNNs):
Introduction & model variants

2) GNNs for unsupervised learning
with graph structured data



3) Modeling implicit structure
with Neural Relational Inference



5) Outlook & Conclusion

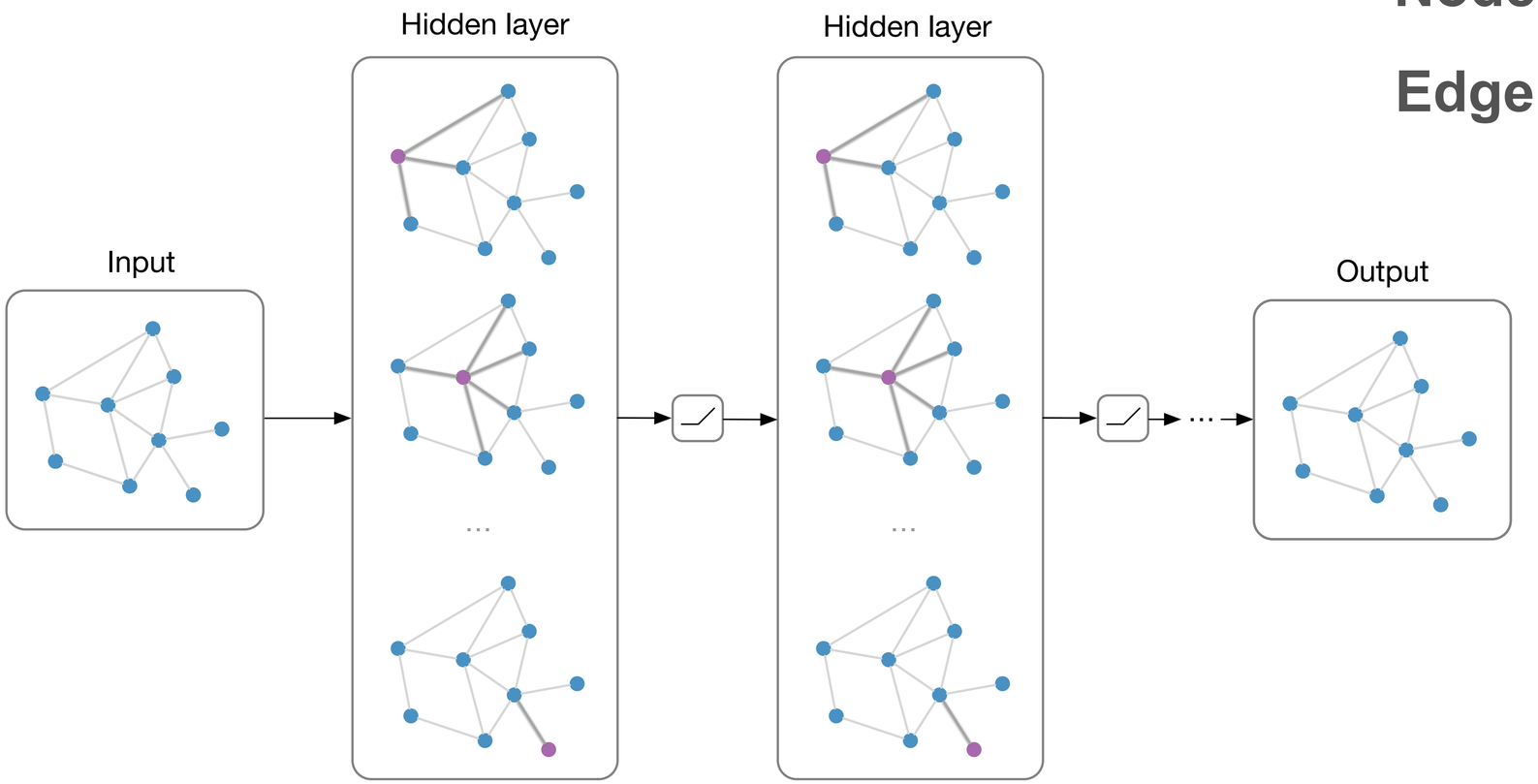
Graph Neural Networks (GNNs)

The bigger picture:

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Node features $\mathbf{h}_i$

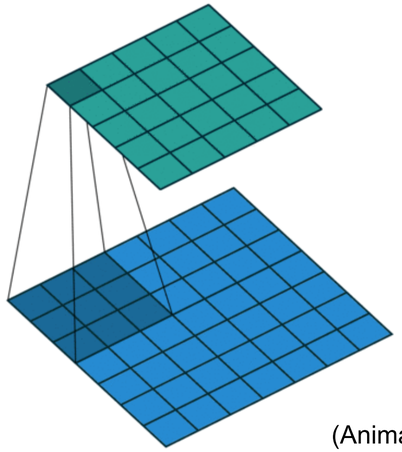
Edge features $\mathbf{h}_{(i,j)}$



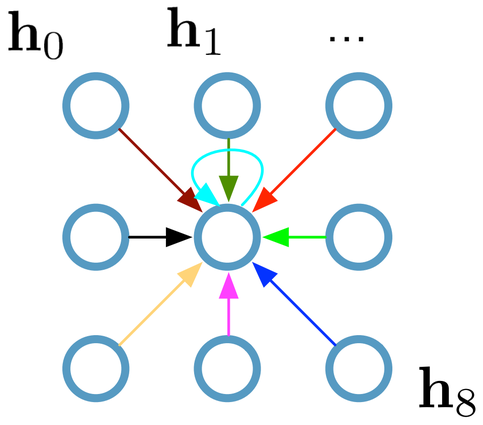
Main idea: Pass messages along edges of graph, agglomerate & transform

CNNs (on grids) as message passing

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

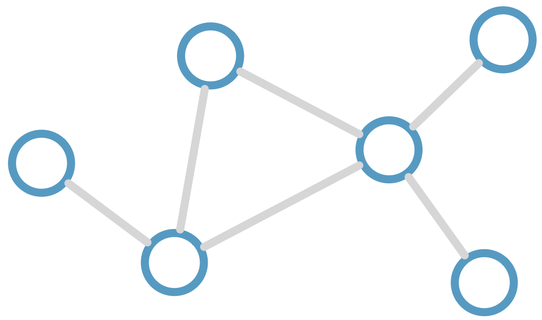
Full update:

$$\mathbf{h}'_4 = \sigma(\mathbf{W}_0\mathbf{h}_0 + \mathbf{W}_1\mathbf{h}_1 + \cdots + \mathbf{W}_8\mathbf{h}_8)$$

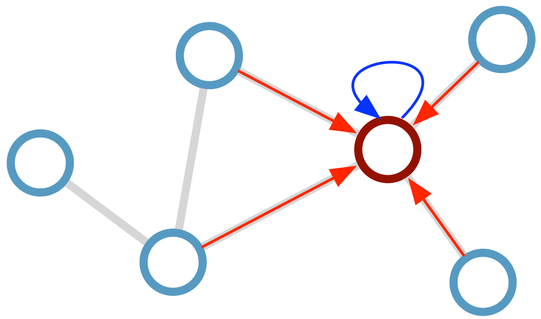
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), based on work by Gori et al. (2005), Bruna et al. (ICLR 2015), Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i' = \sigma\left(\mathbf{W}_0\mathbf{h}_i + \sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}_1\mathbf{h}_j\right)$$

Desirable properties:

- Weight sharing over all locations
- Linear complexity O(E)
- Applicable in inductive settings (new nodes/edges)

Limitations:

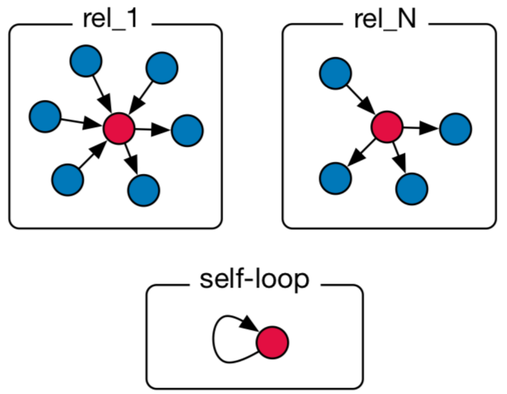
- Does not consider pairwise interactions
- No support for edge features

$\mathcal{N}_i$: neighbor indices α_{ij} : norm. constant

More expressive GNN variants

MoNet & Relational GCN for modeling relational data

Monti et al. (CVPR 2017), Schlichtkrull & Kipf et al. (arXiv 2017, ESWC 2018)



$$\mathbf{h}_i' = \sigma\left(\sum_{r=1}^{R}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{r}\mathbf{W}_r\mathbf{h}_j\right)$$

α_{ij}^{r} **based on:**

- Edge type (Relational GCN)
- Auxiliary features (MoNet), e.g. node degree

Self-attention and Graph Attention Networks

Vaswani et al. (NIPS 2017), Veličković et al. (ICLR 2018)

$$\alpha_{ij}^{r} = \frac{\exp(\mathbf{h}_i^T\mathbf{W}_r'\mathbf{h}_j)}{\sum_{k\in\mathcal{N}_i}\exp(\mathbf{h}_i^T\mathbf{W}_r'\mathbf{h}_k)}$$

Multi-head dot product attention:

- Activation-dependent α_{ij}^{r}
- Bi-linear scoring

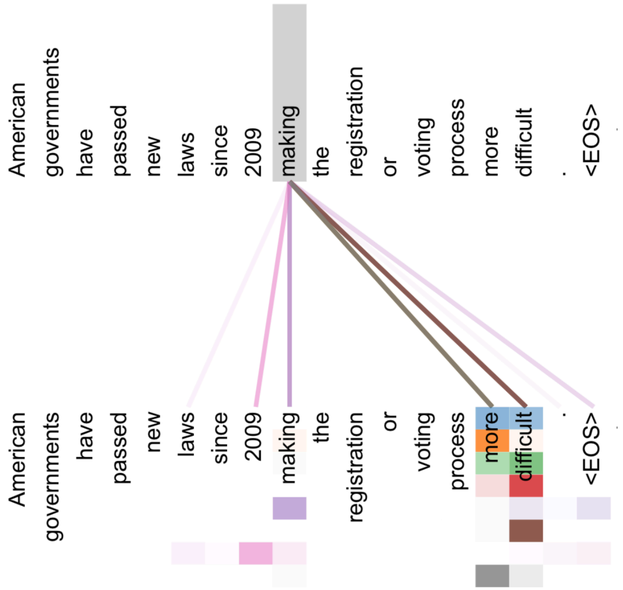
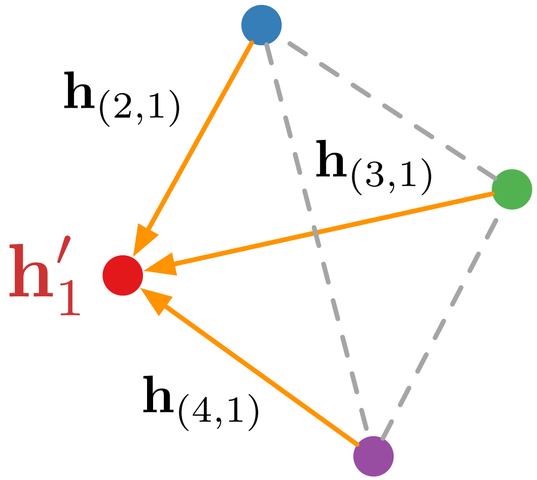


Figure from Vaswani et al. (NIPS 2017)

GNNs with edge embeddings (Neural message passing)

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



Formally:

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

$$e \to v: \quad \mathbf{h}'_j = f_v\left(\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}\right)$$

Pros:

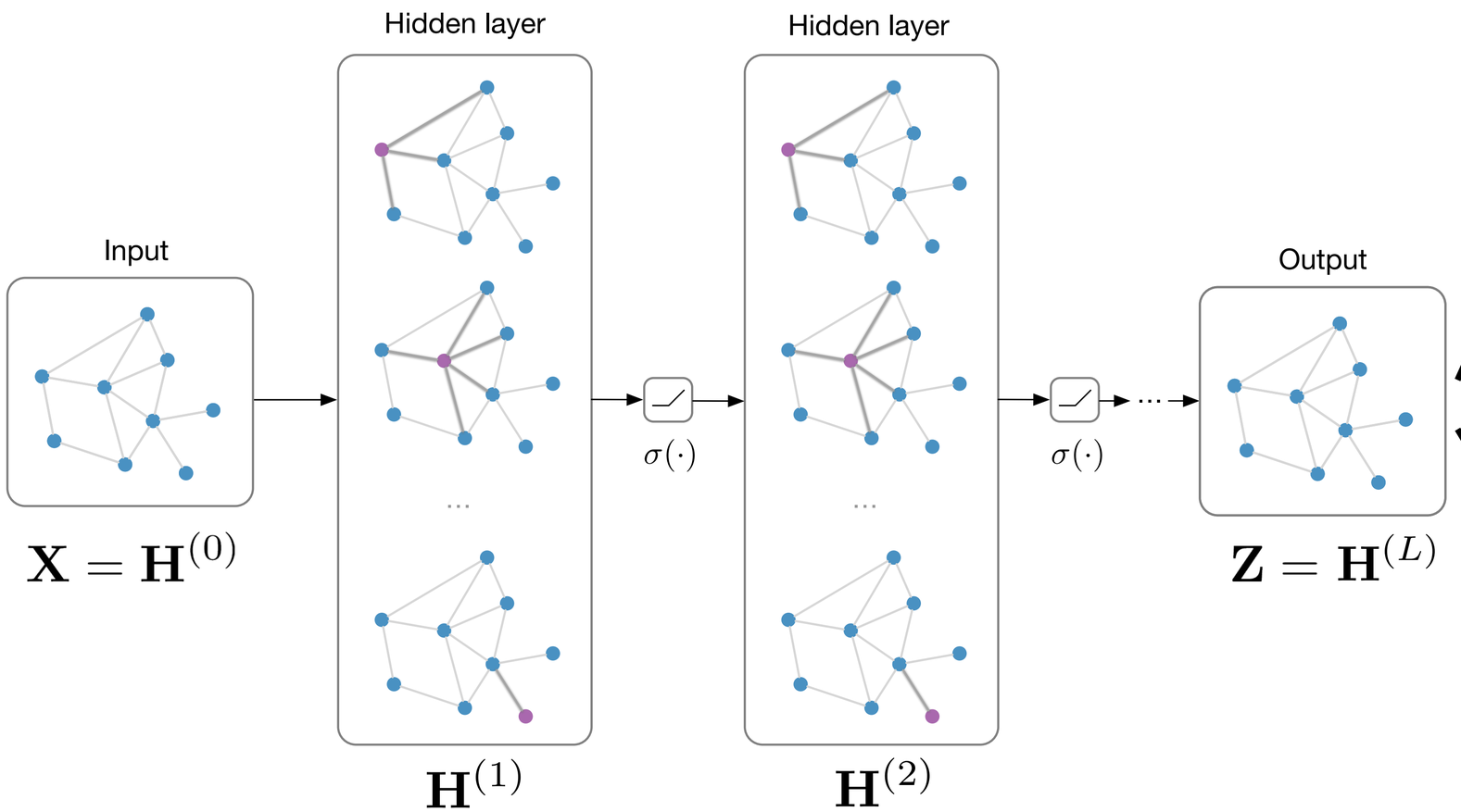
- Support for edge features
- Very flexible / expressive parameterization
- Supports sparse ops

Cons:

- Need to store intermediate edge-based activations
- In practice significantly slower than GCN / self-attention models

GNNs for (semi-)supervised learning on graphs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$, graph adjacency matrix $\mathbf{A}$



Node classification:

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

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

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

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"

Unsupervised learning with GNNs

Objective: Learn node embeddings for downstream tasks

Most approaches follow a contrastive learning approach:

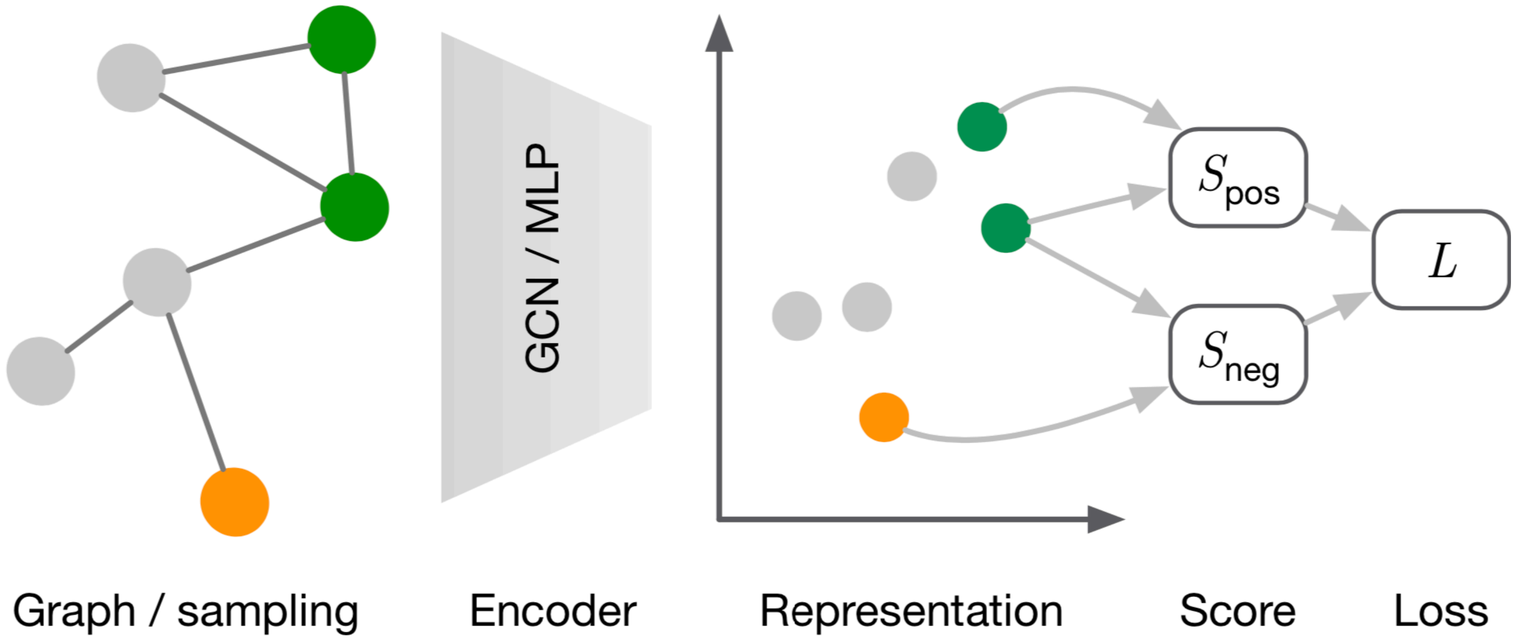


Figure from: Daza & Kipf, A modular framework for unsupervised graph representation learning *(work in progress)*

Unsupervised learning with GNNs

Objective: Learn node embeddings for downstream tasks

Most approaches follow a contrastive learning approach:

- Sampling strategies
  (e.g. pos: neighbor; neg: random node)
- Encoder variants
  (GCN, GAT, MLP, Lookup table)
- Node representations
  (Geometry of latent space, distributional embeddings)
- Score functions (Energy-based, inner / bilinear product, local vs. global)
- Loss (Cross-entropy, hinge, square-exponential)

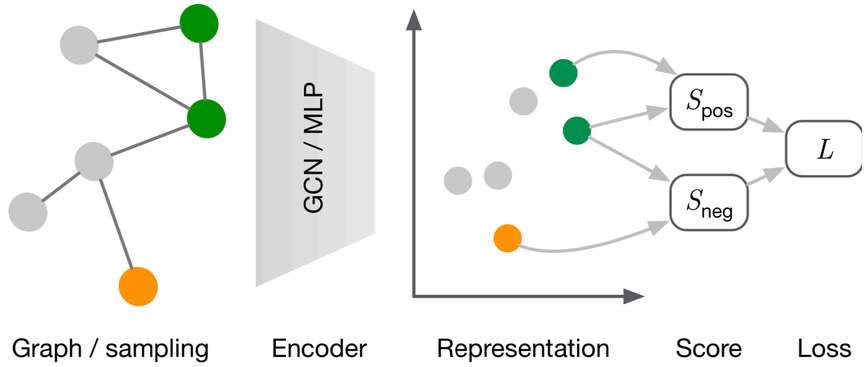


Figure from: Daza & Kipf, A modular framework for unsupervised graph representation learning *(work in progress)*

Examples in literature

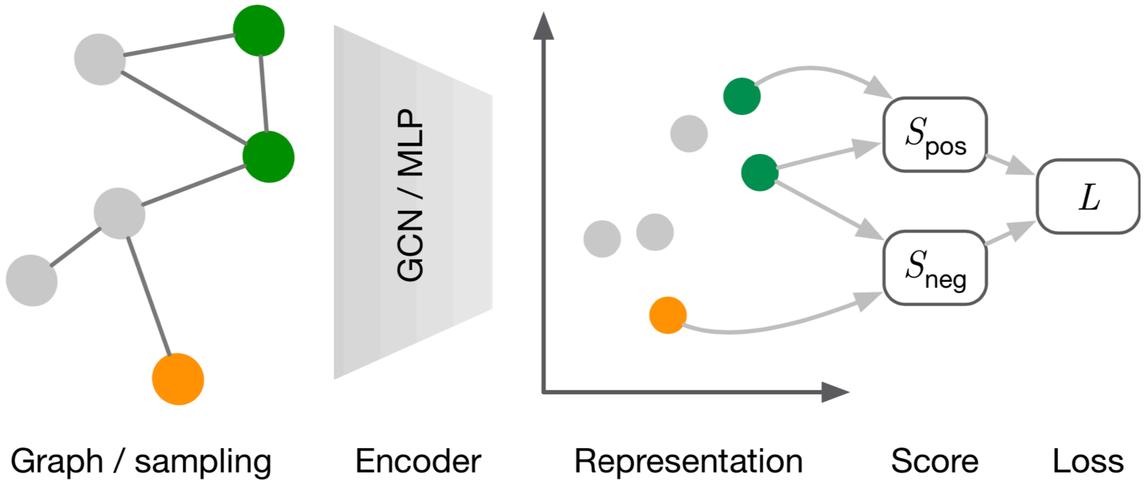
Table 1. Components of unsupervised learning methods on graphs in existing algorithms: DeepWalk (Perozzi et al., 2014), GAE (Kipf & Welling, 2016b), $\mathcal{S}$-VGAE (Davidson et al., 2018), DGI (Veličković et al., 2018), and G2G (Bojchevski & Günnemann, 2017).

Method	Encoder	Representation	Score	Loss	Sampling
DeepWalk	LUT	$\mathbf{z}_i \in \mathbb{R}^D$	$\sigma(\mathbf{z}_i^\top \mathbf{z}_j)$	$-\log S - \log(1-\tilde{S})$	(+) random walk neighbors (-) non-neighbors
GAE	GCN	$\mathbf{z}_i \in \mathbb{R}^D$	$\sigma(\mathbf{z}_i^\top \mathbf{z}_j)$	$-\log S - \log(1-\tilde{S})$	(+) 1st order neighbors (-) non-neighbors
$\mathcal{S}$-VGAE	GCN	$\mathbf{z}_i \sim \text{vMF}(\mathbf{z})$	$\sigma(\mathbf{z}_i^\top \mathbf{z}_j)$	$-\log S - \log(1-\tilde{S})$	(+) 1st order neighbors (-) non-neighbors
DGI	GCN	$\mathbf{z}_i \in \mathbb{R}^D$	$\sigma\left(\mathbf{z}_i^T \mathbf{W} \mathbf{s}\right)$	$-\log S - \log(1-\tilde{S})$	(+) original graph (-) corrupted graph
G2G	MLP	$\mathbf{z}_\mu \in \mathbb{R}^D$ $\mathbf{z}_\Sigma \in \mathbb{R}^D$	$\exp(-\text{KL}(\mathcal{N}_i \| \mathcal{N}_j))$	$(\log S)^2 + \tilde{S}$	(+) 1st order neighbors (-) higher order neighbors

GAE: Graph Auto-Encoders, G2G: Graph2Gauss, DGI: Deep Graph Infomax $\mathbf{s} = \sum_{i=1}^{N} \mathbf{z}_i$

Daza & Kipf, A modular framework for unsupervised graph representation learning *(work in progress)*

(Early) take-aways for unsupervised learning



- Graph-based encoders often improve performance
- Neighbor-based scoring (GAE) effective for both link prediction & node classification
- Local-global scoring (DGI) especially effective for node classification
- Ideal node representation (distributional, hyperspherical, etc.) heavily data-dependent

Daza & Kipf, A modular framework for unsupervised graph representation learning *(work in progress)*

ModGraph: PyTorch library — to be released soon

Graph Autoencoders (Kipf and Welling, 2016):

```
encoder = GCNEncoder(dataset.num_features, hidden_dims=[256, 128])
representation = EuclideanInnerProduct()
loss = bceloss
sampling = FirstNeighborSampling
```

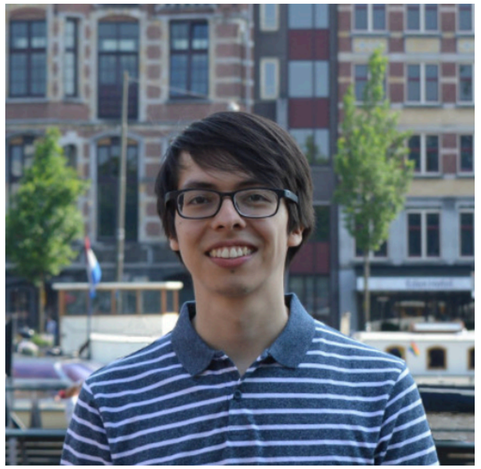
Deep Graph Infomax (Veličković et al., 2018):

```
encoder = GCNEncoder(dataset.num_features, hidden_dims=[256, 128])
representation = EuclideanBilinear()
loss = bceloss
sampling = GraphCorruptionSampling
```

Graph2Gauss (Bojchevski and Günnemann, 2017):

```
encoder = MLPEncoder(dataset.num_features, hidden_dims=[256, 128])
representation = Gaussian()
loss = square_exponential
sampling = RankedSampling
```

Developed by
Daniel Daza



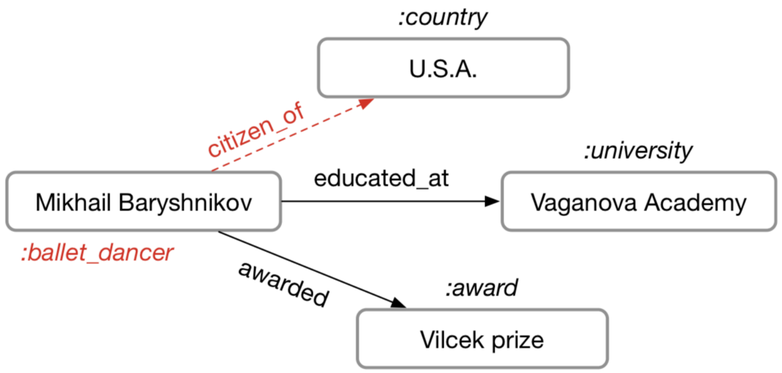
https://dfdazac.github.io

Other GNN/GCN applications

Recommender systems

Ying et al., (2018); vd Berg, Kipf & Welling (2017), Monti et al. (2017)

Knowledge base completion

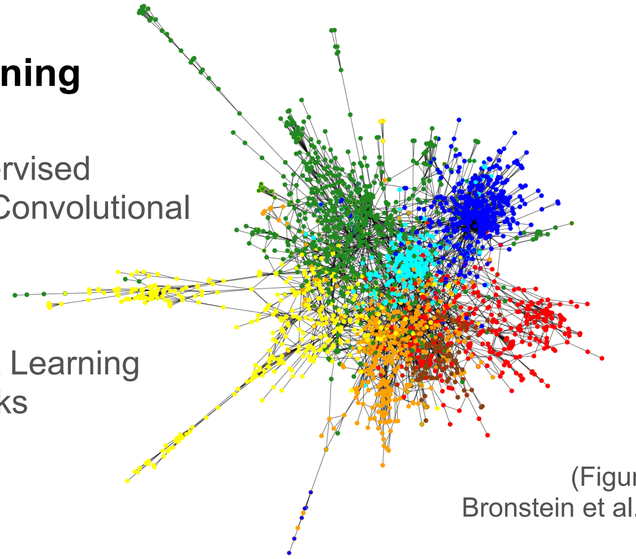


Schlichtkrull & Kipf et al. (ESWC 2018)

Semi-supervised learning

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

Garcia & Bruna, Few-Shot Learning with Graph Neural Networks (ICLR 2018)



(Figure from: Bronstein et al., 2016)

Visual question answering

Norcliffe-Brown et al., Learning Conditioned Graph Structures for VQA, (NeurIPS 2018)

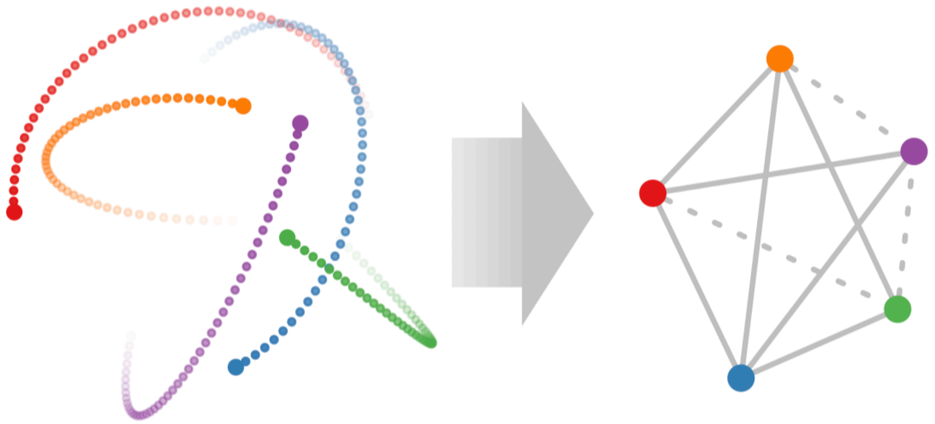
Q: Are all these animals the same colour?

Prediction: Yes

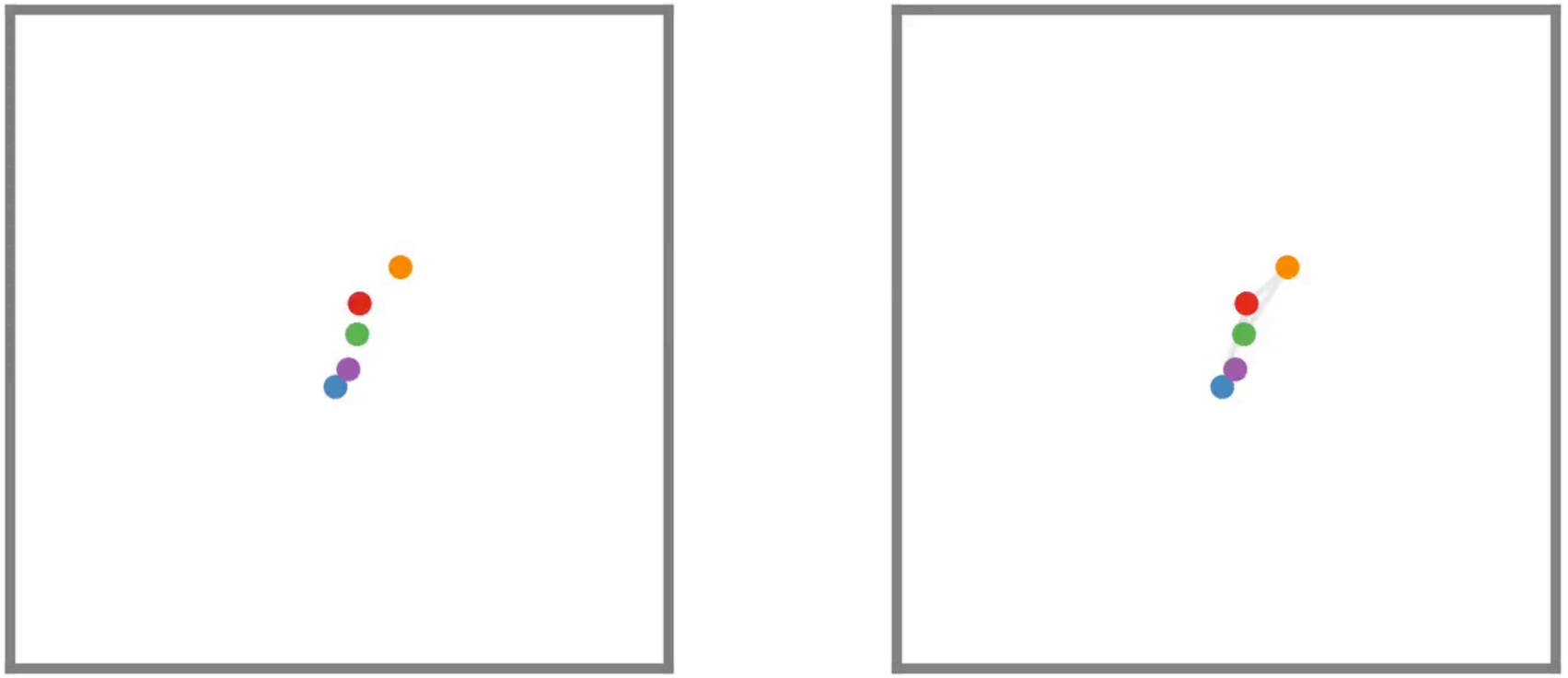
Answer: Yes

Modeling implicit / hidden structure

with Neural Relational Inference

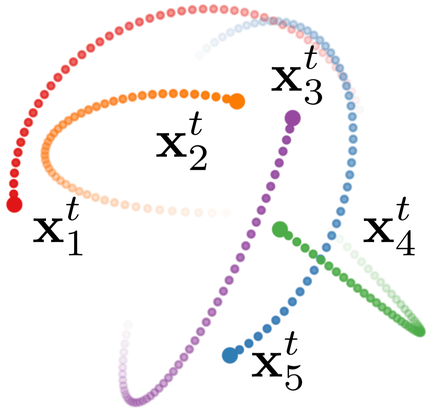


Motivation: Learning multi-object dynamics



Motivation: Learning multi-object dynamics

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



Observed dynamics

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

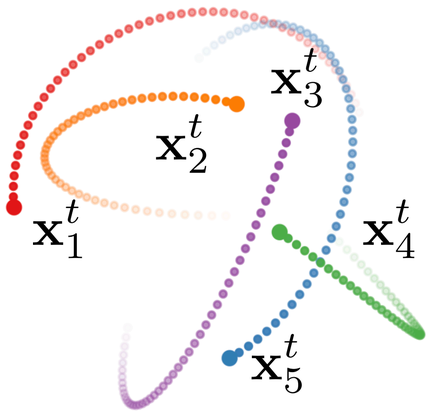
$$\mathbf{x}^t = [\mathbf{x}_1^t, \mathbf{x}_2^t, ..., \mathbf{x}_5^t]$$

- Feed into neural net

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

- Done?

Motivation: Learning multi-object dynamics



Observed dynamics

Problems:

- Arbitrary ordering of nodes

  ➡ Need permutation invariance

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

  ➡ Often, interactions are **pairwise**

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

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

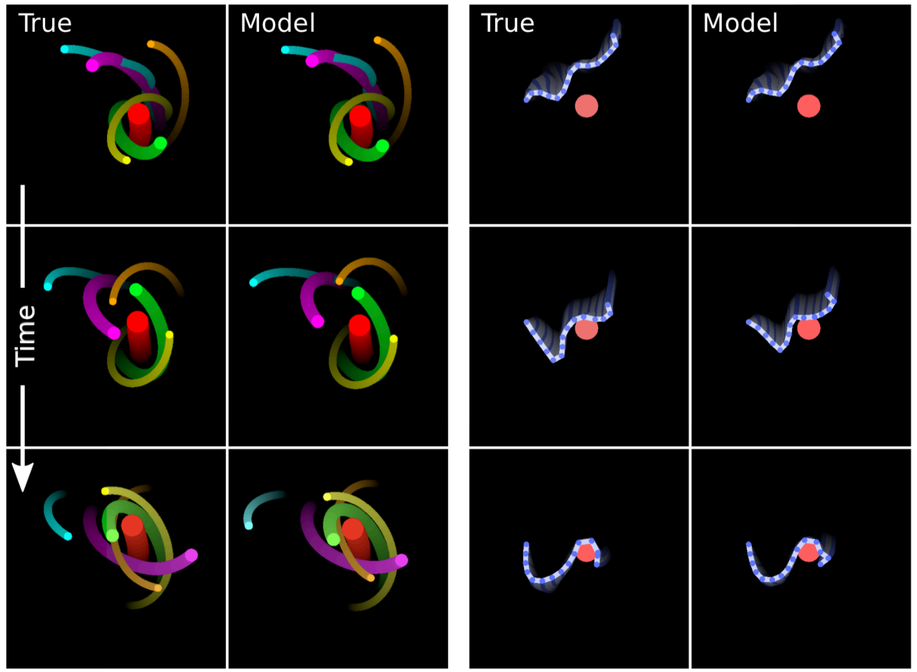
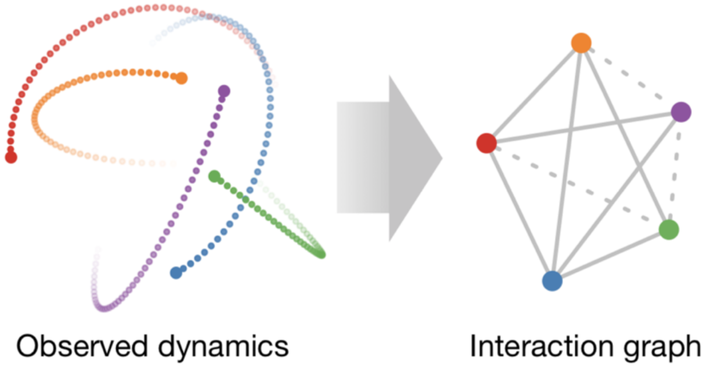


Figure from Battaglia et al. (NIPS 2016)

But what if we don't know the interactions?

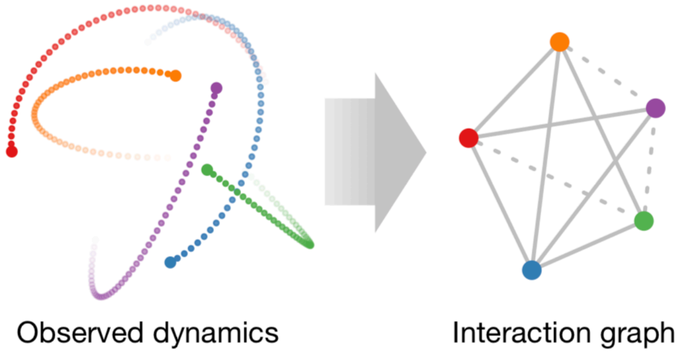


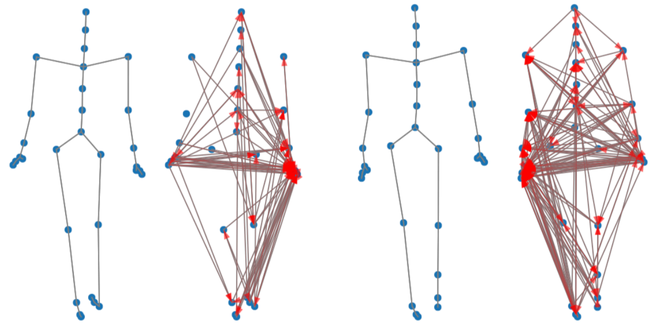
Neural Relational Inference for Interacting Systems

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

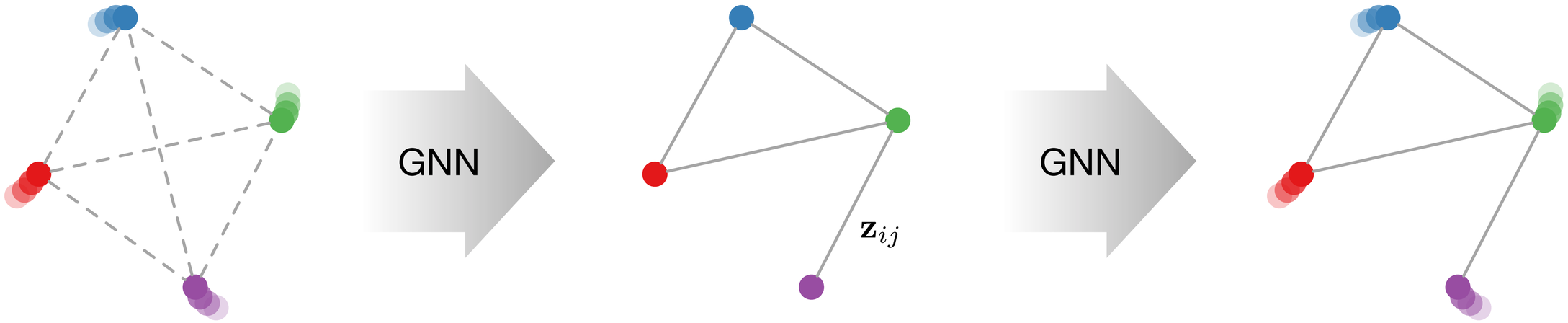
Our work (ICML 2018)**:**

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





Neural Relational Inference with Graph Neural Networks



Observe dynamics

$\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$

Interaction graph proposal

$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

Reconstruct dynamics*

$p_\theta(\mathbf{x}|\mathbf{z})$

VAE Objective (ELBO) $\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]$

Reconstruction **Regularization**

* or predict future dynamics; technically, we have to condition decoder on at least one time step of **x**

Encoder in detail

Graph Neural Net on full graph:

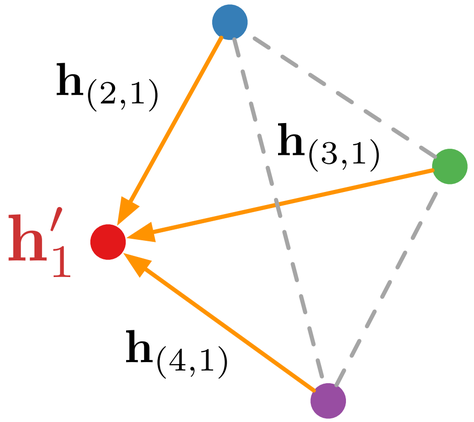
$$\mathbf{h}_j = f_{\text{emb}}(\mathbf{x}_j)$$

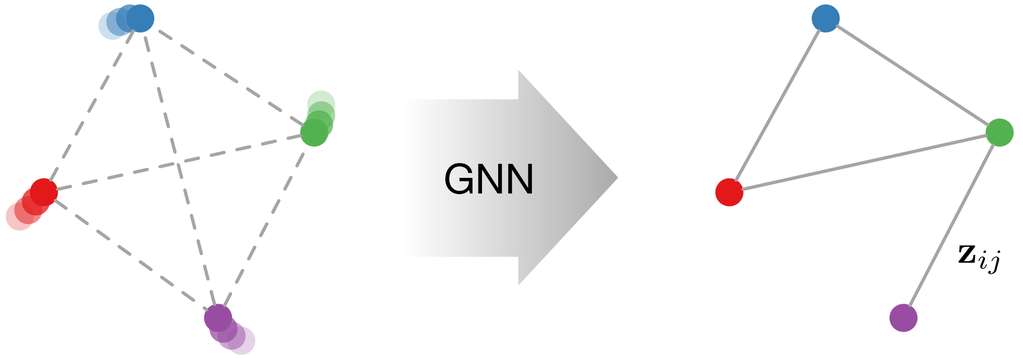
$$v \to e: \quad \mathbf{h}_{(i,j)} = f_e^1([\mathbf{h}_i, \mathbf{h}_j])$$

$$e \to v: \quad \mathbf{h}'_j = f_v^1\left(\sum_{i \neq j} \mathbf{h}_{(i,j)}\right)$$

$$v \to e: \quad \mathbf{h}'_{(i,j)} = f_e^2([\mathbf{h}'_i, \mathbf{h}'_j])$$

$e \to v$:





Observe dynamics

$\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$

Interaction graph proposal

$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

What about z?

$\mathbf{z} \in \mathbb{R}^{N \times N \times K}$: binary adjacency tensor

$\mathbf{z}_{ij}$: edge type (one-hot vector)

$$\mathbf{z}_{ij} \sim \text{Categorical}\left(\text{softmax}(\mathbf{h}'_{(i,j)})\right)$$

Sampling **z** at training time

During training, use samples from relaxed categorical distribution

(concrete / Gumbel softmax*)

$$\mathbf{z}_{ij} = \text{softmax}((\mathbf{h}'_{(i,j)} + \mathbf{g})/\tau)$$

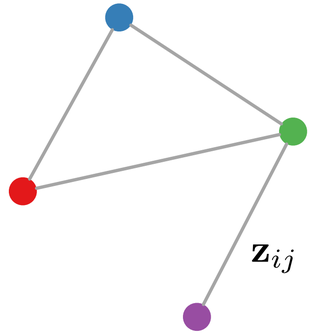
Gumbel noise:

$\mathbf{g} \in \mathbb{R}^K$ with $g_k \sim \text{Gumbel}(0, 1)$

Temperature: τ (e.g. 0.5)

➡ Gradients via reparameterization trick

(Kingma & Welling, ICLR 2014)



Interaction graph proposal

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$$

Prior $p(\mathbf{z}) = \prod_{i,j \neq i} p(\mathbf{z}_{ij})$ **:**

1) Uniform categorical distribution

2) Sparsity-inducing prior

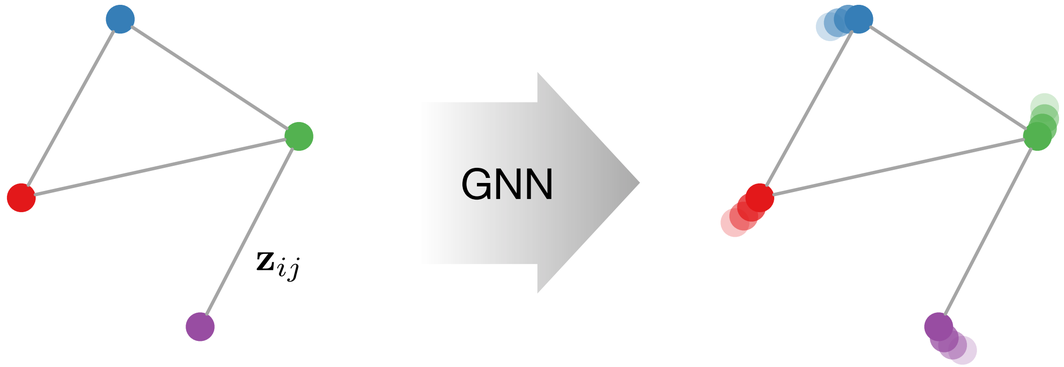
(if one edge type is hard-coded as “no interaction”)

* Jang et al. (ICLR 2017), Maddison et al. (ICLR 2017)

Decoder in detail

Main idea:

Run GNN on latent graph **z**, predicted by encoder.



Interaction graph proposal

$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

Reconstruct dynamics

$p_\theta(\mathbf{x}|\mathbf{z})$

For Markovian dynamics:

$$v \to e: \quad \tilde{\mathbf{h}}^t_{(i,j)} = \sum_k z_{ij,k} \tilde{f}^k_e([\mathbf{x}^t_i, \mathbf{x}^t_j])$$

Weigh messages by edge type

$$e \to v: \quad \boldsymbol{\mu}^{t+1}_j = \mathbf{x}^t_j + \tilde{f}_v(\textstyle\sum_{i \neq j} \tilde{\mathbf{h}}^t_{(i,j)})$$

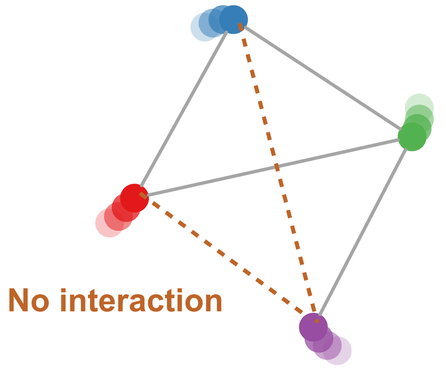
Predict location difference

$$p(\mathbf{x}^{t+1}_j | \mathbf{x}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}^{t+1}_j, \sigma^2 \mathbf{I})$$

Predictive distribution (fixed variance)

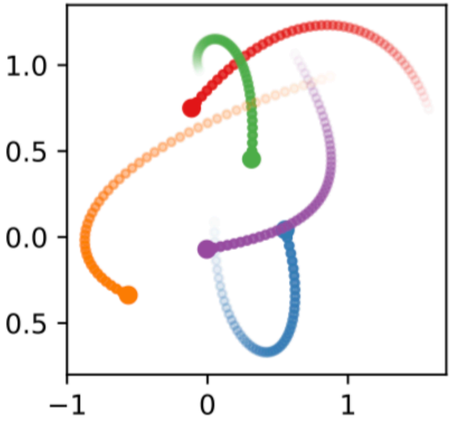
Decoder variants

- Teacher forcing only every M-th time step (e.g. M = 10)
- Hard-code certain edge types (e.g. “no interaction”)
- **Recurrent decoder** (non-Markovian dynamics)
- (Differentiable) **Physics simulation decoder**
- **True graph** (if known) as input to decoder
- **Fully-connected graph** as input to decoder
- **Dynamic graph** re-estimation at test time



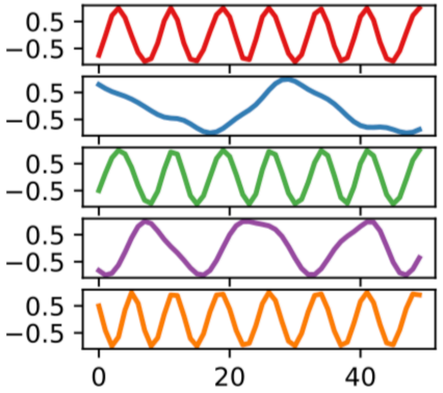
Reconstruct dynamics

Environments



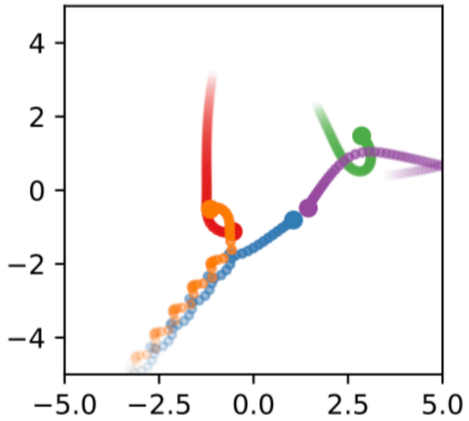
Balls connected by springs

$$\mathbf{F}_{ij} = -C(\mathbf{x}_i - \mathbf{x}_j)$$



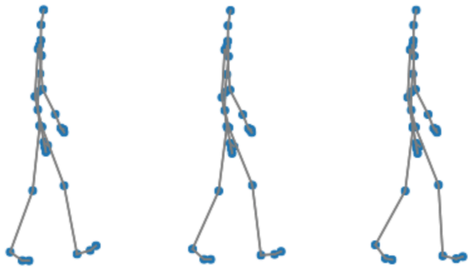
Phase-coupled oscillators (Kuramoto model)

$$\frac{d\phi_i}{dt} = \omega_i + \sum_{j \neq i} k_{ij} \sin(\phi_i - \phi_j)$$

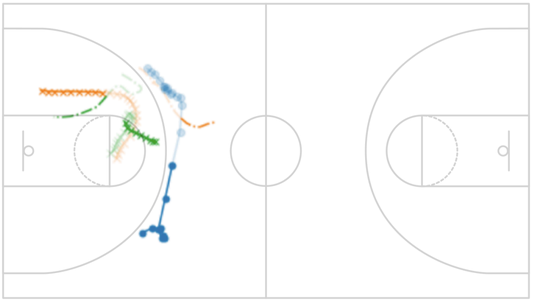


Charged particles

$$\mathbf{F}_{ij} = C \cdot \text{sign}(q_i \cdot q_j) \frac{\mathbf{x}_i - \mathbf{x}_j}{||\mathbf{x}_i - \mathbf{x}_j||^3}$$



Motion capture

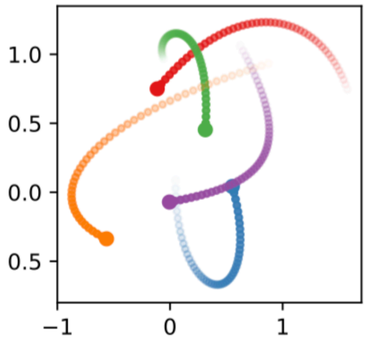


NBA sports tracking

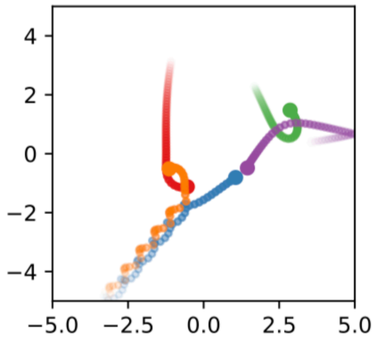
Learning latent interaction graphs

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

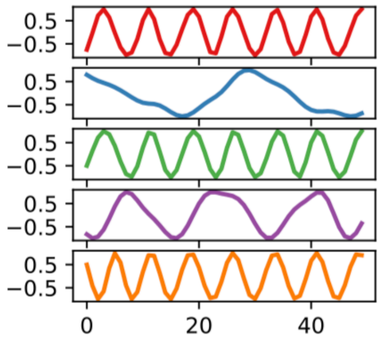
Model	Springs	Charged	Kuramoto
	5 objects		
Corr. (path)	52.4±0.0	55.8±0.0	62.8±0.0
Corr. (LSTM)	52.7±0.9	54.2±2.0	54.4±0.5
NRI (sim.)	**99.8**±0.0	59.6±0.8	–
NRI (learned)	**99.9**±0.0	**82.1**±0.6	**96.0**±0.1
Supervised	99.9±0.0	95.0±0.3	99.7±0.0



Springs (2D)



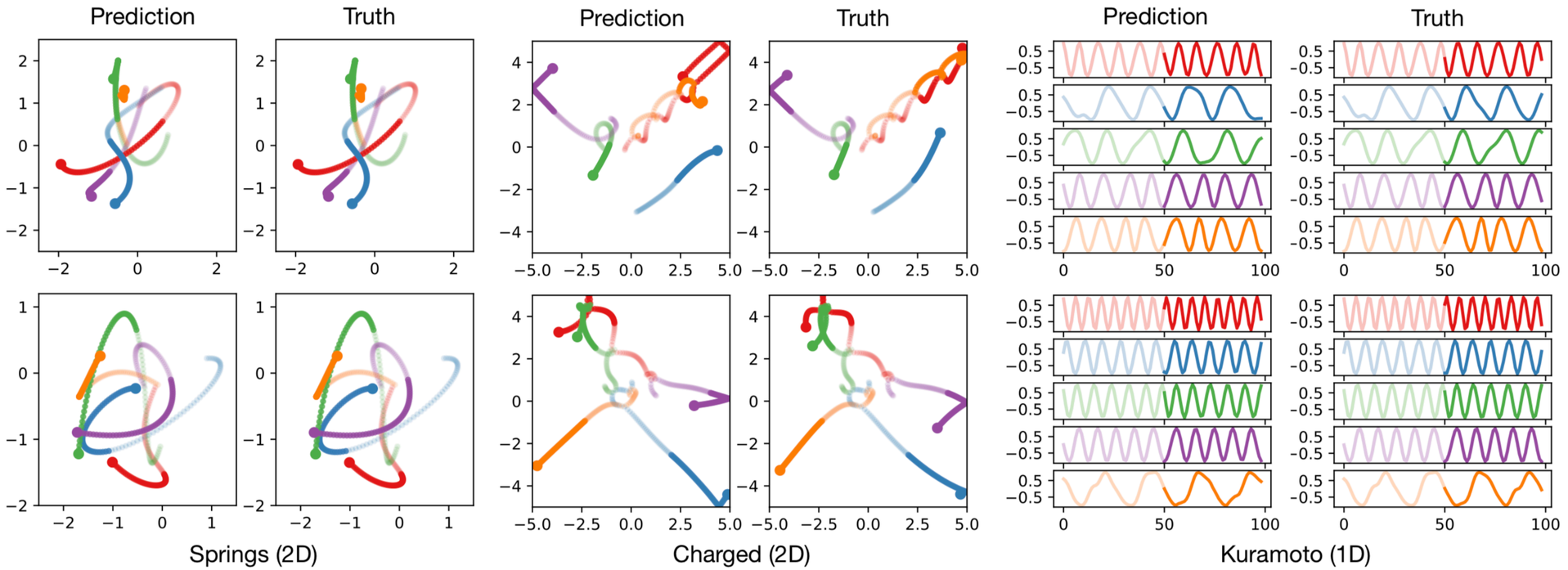
Charged (2D)

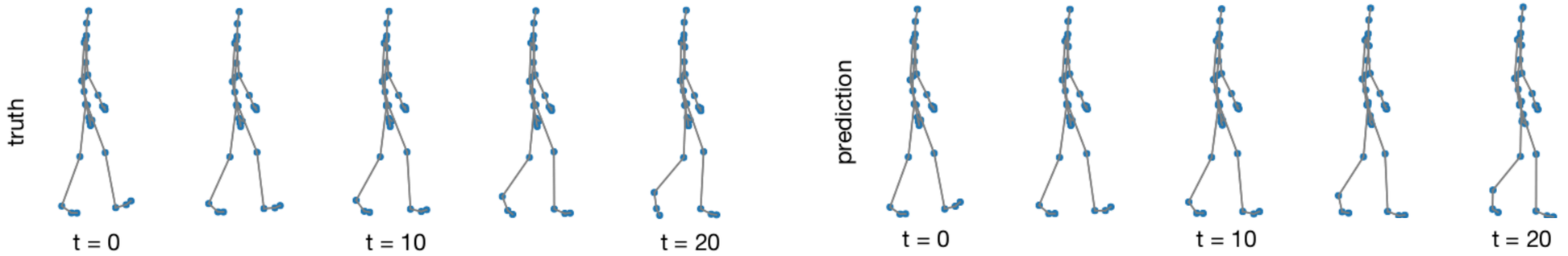


Kuramoto (1D)

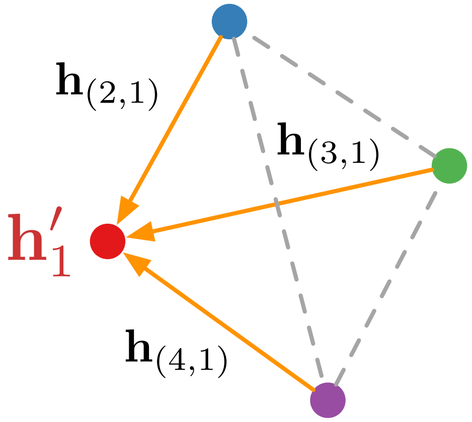
NRI can learn to discover ground-truth relations with very high accuracy!

Qualitative results





Summary and outlook

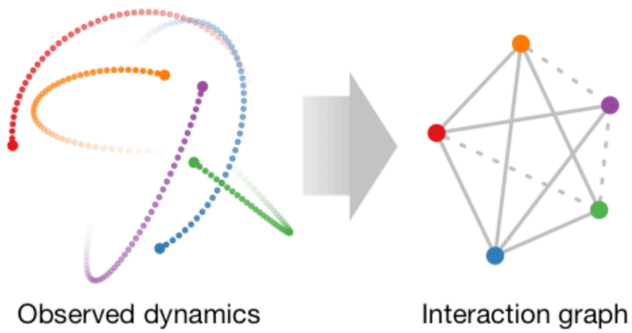


- **GNNs** are a powerful class of models for data with explicit or implicit relational structure
- Exciting new research field with massive growth (methods and applications) in past 3 years

Next steps:

- Systematic exploration of GNN variants
- Joint discovery of entities / objects and relational structure from raw data
- Other structure-informed inductive biases: e.g., compositionality & hierarchy

Further reading



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)

Blog post: https://medium.com/@tkipf

Other material:

Blog post on GCNs
https://tkipf.github.io/graph-convolutional-networks

Code on Github
http://github.com/tkipf

Special thanks to collaborators: Ethan Fetaya, Daniel Daza, Rianne van den Berg, Michael Schlichtkrull, Jackson Wang, Peter Bloem, Jakub Tomczak, Ivan Titov, Max Welling, Richard Zemel and others

Project supported by SAP