Structured deep models:
Deep learning on graphs and beyond

Thomas Kipf, 25 May 2018
CompBio Seminar, University of Cambridge

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

IMAGENET

Speech data

Natural language processing (NLP)

... The cat sat on the mat.

Grid games
Deep neural nets that exploit:
- translation equivariance (weight sharing)
- hierarchical compositionality

Natural language processing (NLP):
- Sentence
  - Predicate / Verb Phrase
  - Prepositional Phrase
  - Noun Phrase
  - Article
  - Noun
  - Verb
  - Preposition

Speech data

Grid games

IMAGENET

The Deep Learning slide
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
Graph-structured data

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

- Social networks
- Citation networks
- Communication networks
- Multi-agent systems

Protein interaction networks

Molecules
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

- Protein interaction networks

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
- Protein interaction networks
- Knowledge graphs
- Molecules
- Road maps

Structured Deep Models

Thomas Kipf
Graph-structured data

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

Social networks
Citation networks
Communication networks
Multi-agent systems

Protein interaction networks

Knowledge graphs

Molecules

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**

   - Node classification
   - Graph classification
   - Link prediction
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)

Main idea: Pass messages between pairs of nodes & agglomerate
Graph Neural Networks (GNNs)

**The bigger picture:**

**Notation:** \( \mathcal{G} = (A, X) \)
- Adjacency matrix \( A \in \mathbb{R}^{N \times N} \)
- Feature matrix \( X \in \mathbb{R}^{N \times F} \)

**Main idea:** Pass messages between pairs of nodes & agglomerate
Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:
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:

\[ h_i \in \mathbb{R}^F \] are (hidden layer) activations of a pixel/node
Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:

Update for a single pixel:
- Transform messages individually $W_i h_i$
- Add everything up $\sum_i W_i h_i$

$h_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node
Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:

Update for a single pixel:
- Transform messages individually: $W_i h_i$
- Add everything up: $\sum_i W_i h_i$

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

Full update:

$$h_4^{(l+1)} = \sigma \left( W_0^{(l)} h_0^{(l)} + W_1^{(l)} h_1^{(l)} + \cdots + W_8^{(l)} h_8^{(l)} \right)$$
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:
Consider this undirected graph:

Calculate update for node in red:

Update rule:

$$h_i^{(l+1)} = \sigma \left( h_i^{(l)} W_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{C_{ij}} h_j^{(l)} W_1^{(l)} \right)$$

$\mathcal{N}_i$: 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:

\[
 h_i^{(l+1)} = \sigma \left( h_i^{(l)} W_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l)} W_1^{(l)} \right)
\]

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

\( \mathcal{N}_i \): neighbor indices \quad c_{ij} : \text{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:

$$h_i^{(l+1)} = \sigma \left( h_i^{(l)}W_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{C_{ij}} h_j^{(l)}W_1^{(l)} \right)$$

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

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

Update rule:

$$h_i^{(l+1)} = \sigma \left( h_i^{(l)} W_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l)} W_1^{(l)} \right)$$

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

$\mathcal{N}_i$: 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:

\[ v \rightarrow e : \quad h_{(i,j)}^l = f_e^l([h_i^l, h_j^l, x_{(i,j)}]) \]

\[ e \rightarrow v : \quad h_j^{l+1} = f_v^l([\sum_{i \in \mathcal{N}_j} h_{(i,j)}^l, x_j]) \]
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:
\[
\begin{align*}
  v \rightarrow e & : \quad h_{(i,j)}^{l} = f_e^l([h_i^l, h_j^l, x_{(i,j)}]) \\
  e \rightarrow v & : \quad h_j^{l+1} = f_v^l([\sum_{i \in N_j} h_{(i,j)}^l, x_j])
\end{align*}
\]
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

Cons:
- Need to store intermediate edge-based activations
- Difficult to implement with subsampling
  ➡ In practice limited to small graphs

Formally:
\[
\begin{align*}
\forall v \rightarrow e & : \quad h_{(i,j)}^l = f_e^l([h_i^l, h_j^l, x_{(i,j)}]) \\
\forall e \rightarrow v & : \quad h_j^{l+1} = f_v^l([\sum_{i \in N_j} h_{(i,j)}^l, x_j])
\end{align*}
\]
Graph neural networks with attention
Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)

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

[Figure from Veličković et al. (ICLR 2018)]
Graph neural networks with attention

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

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

\[
\alpha_{ij} = \frac{\exp \left( \text{LeakyReLU} \left( \tilde{a}_i^T [W \tilde{h}_i \| W \tilde{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left( \text{LeakyReLU} \left( \tilde{a}_i^T [W \tilde{h}_i \| W \tilde{h}_k] \right) \right)}
\]
Graph neural networks with attention
Monti et al. (CVPR 2017), Hoshen (NIPS 2017), 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
Graph neural networks with attention
Monti et al. (CVPR 2017), Hoshen (NIPS 2017), 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

\[
\tilde{h}_i' = \sigma \left( \frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{ij}^k W^k \tilde{h}_j \right)
\]
\[
\alpha_{ij} = \frac{\exp \left( \text{LeakyReLU} \left( \tilde{a}_T^T [W \tilde{h}_i \Vert W \tilde{h}_j] \right) \right)}{\sum_{k \in N_i} \exp \left( \text{LeakyReLU} \left( \tilde{a}_T^T [W \tilde{h}_i \Vert W \tilde{h}_k] \right) \right)}
\]
A brief history of graph neural nets

“Spatial methods”

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

GCN
Kipf & Welling (ICLR 2017)

“Spectral methods”

Spectral Graph CNN
Bruna et al. (ICLR 2015)

ChebNet
Defferrard et al. (NIPS 2016)

(Slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

MoNet
Monti et al. (CVPR 2017)

“Spectral methods”

Spectral Graph CNN
Bruna et al. (ICLR 2015)

ChebNet
Defferrard et al. (NIPS 2016)

GCN
Kipf & Welling (ICLR 2017)

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

MoNet
Monti et al. (CVPR 2017)

“Spectral methods”

Spectral Graph CNN
Bruna et al. (ICLR 2015)

ChebNet
Defferrard et al. (NIPS 2016)

Neural MP
Gilmer et al. (ICML 2017)

GCN
Kipf & Welling (ICLR 2017)

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”
- Original GNN  
  Gori et al. (2005)
- GG-NN  
  Li et al. (ICLR 2016)
- MoNet  
  Monti et al. (CVPR 2017)
- Neural MP  
  Gilmer et al. (ICML 2017)
- GraphSAGE  
  Hamilton et al. (NIPS 2017)

“Spectral methods”
- Spectral Graph CNN  
  Bruna et al. (ICLR 2015)
- ChebNet  
  Defferrard et al. (NIPS 2016)
- GCN  
  Kipf & Welling (ICLR 2017)

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

MoNet
Monti et al. (CVPR 2017)

Relation Nets
Santoro et al. (NIPS 2017)

GraphSAGE
Hamilton et al. (NIPS 2017)

“Spectral methods”

GG-NN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

MoNet
Monti et al. (CVPR 2017)

GCN
Kipf & Welling (ICLR 2017)

Neural MP
Gilmer et al. (ICML 2017)

“Spectral methods”

GCN
Kipf & Welling (ICLR 2017)

ChebNet
Defferrard et al. (NIPS 2016)

Relation Nets
Santoro et al. (NIPS 2017)

MoNet
Monti et al. (CVPR 2017)

“Spectral methods”

GCN
Kipf & Welling (ICLR 2017)

ChebNet
Defferrard et al. (NIPS 2016)

GraphSAGE
Hamilton et al. (NIPS 2017)

“Spectral methods”

GCN
Kipf & Welling (ICLR 2017)

ChebNet
Defferrard et al. (NIPS 2016)

GraphSAGE
Hamilton et al. (NIPS 2017)

GCN
Kipf & Welling (ICLR 2017)

ChebNet
Defferrard et al. (NIPS 2016)

GraphSAGE
Hamilton et al. (NIPS 2017)

GCN
Kipf & Welling (ICLR 2017)

ChebNet
Defferrard et al. (NIPS 2016)

GraphSAGE
Hamilton et al. (NIPS 2017)
A brief history of graph neural nets

“Spatial methods”

- Original GNN
  Gori et al. (2005)

- GG-NN
  Li et al. (ICLR 2016)

- MoNet
  Monti et al. (CVPR 2017)

- Neural MP
  Gilmer et al. (ICML 2017)

- Programs as Graphs
  Allamanis et al. (ICLR 2018)

“Spectral methods”

- Spectral Graph CNN
  Bruna et al. (ICLR 2015)

- ChebNet
  Defferrard et al. (NIPS 2016)

- GCN
  Kipf & Welling (ICLR 2017)

“Spectral methods”

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

"Spatial methods"

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

MoNet
Monti et al. (CVPR 2017)

Neural MP
Gilmer et al. (ICLR 2017)

Relation Nets
Santoro et al. (ICML 2018)

Programs as Graphs
Allamanis et al. (ICLR 2018)

NRI
Kipf et al. (ICML 2018)

"Spectral methods"

Spectral
Bruna et al. (ICLR 2015)

Graph CNN
Bruna et al. (ICLR 2015)

ChebNet
Defferrard et al. (NIPS 2016)

GCN
Kipf & Welling (ICLR 2017)

"Spectral methods"

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”

Original GNN
Gori et al. (2005)

GG-NN
Li et al. (ICLR 2016)

MoNet
Monti et al. (CVPR 2017)

Neural MP
Gilmer et al. (ICLR 2017)

Relation Nets
Santoro et al. (ICLR 2018)

GraphSAGE
Hamilton et al. (NIPS 2017)

Programs as Graphs
Allamanis et al. (ICLR 2018)

NRI
Kipf et al. (ICML 2018)

GAT
Veličković et al. (ICLR 2018)

“Spectral methods”

Spectral
Graph CNN
Bruna et al. (ICLR 2015)

ChebNet
Defferrard et al. (NIPS 2016)

GCN
Kipf & Welling (ICLR 2017)

“Spectral methods”

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”

- Original GNN: Gori et al. (2005)
- GG-NN: Li et al. (ICLR 2016)
- Neural MP: Gilmer et al. (ICLR 2017)
- GCN: Kipf & Welling (ICLR 2017)
- Spectral Graph CNN: Bruna et al. (ICLR 2015)
- ChebNet: Defferrard et al. (NIPS 2016)

“Spectral methods”

- MoNet: Monti et al. (CVPR 2017)
- GraphSAGE: Hamilton et al. (ICLR 2017)
- Programs as Graphs: Allamanis et al. (ICLR 2018)
- MoNet: Monti et al. (CVPR 2017)
- GAT: Veličković et al. (ICL 2018)
- NRI: Kipf et al. (ICML 2018)

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

“Spatial methods”
- Original GNN: Gori et al. (2005)
- GG-NN: Li et al. (ICLR 2016)
- Neural MP: Gilmer et al. (ICML 2017)
- GCN: Kipf & Welling (ICLR 2017)
- Spectral Graph CNN: Brun et al. (ICLR 2015)
- ChebNet: Defferrard et al. (NIPS 2016)

“Spectral methods”
- MoNet: Monti et al. (CVPR 2017)
- Spectral methods
- “DL on graph explosion”
- GraphSAGE: Hamilton et al. (NIPS 2017)
- Relation Nets: Santoro et al. (ICLR 2018)
- Programs as Graphs: Allamanis et al. (ICML 2018)
- NRI: Kipf et al. (ICML 2018)
- GAT: Veličković et al. (ICLR 2018)

(slide inspired by Alexander Gaunt’s talk on GNNs)
A brief history of graph neural nets

**“Spatial methods”**
- Original GNN: Gori et al. (2005)
- GG-NN: Li et al. (ICLR 2016)
- GCN: Kipf & Welling (ICLR 2017)
- Spectral Graph CNN: Bruna et al. (ICLR 2015)
- ChebNet: Defferrard et al. (NIPS 2016)

**“Spectral methods”**
- MoNet: Monti et al. (CVPR 2017)
- Neural MP: Gilmer et al. (ICML 2017)
- Spectral methods: “DL on graph explosion”
- GraphSAGE: Hamilton et al. (NIPS 2017)
- Programs as Graphs: Allamanis et al. (ICLR 2018)
- Relation Nets: Santoro et al. (ICLR 2018)
- NRI: Kipf et al. (ICML 2018)
- GAT: Veličković et al. (ICLR 2018)

**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 $X \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{A}$

\[ X = H^{(0)} \]

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

\[ Z = H^{(N)} \]
One fits all: Classification and link prediction with GNNs/GCNs

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

**Node classification:**
$$\text{softmax}(\mathbf{z}_n)$$

*e.g. Kipf & Welling (ICLR 2017)*

\[
\mathbf{X} = \mathbf{H}^{(0)} \\
\mathbf{H}^{(l+1)} = \sigma \left( \hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right) \\
\mathbf{Z} = \mathbf{H}^{(N)}
\]
One fits all: Classification and link prediction with GNNs/GCNs

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

- **Node classification:**
  $$\text{softmax}(z_n)$$
  e.g. Kipf & Welling (ICLR 2017)

- **Graph classification:**
  $$\text{softmax}(\sum_n z_n)$$
  e.g. Duvenaud et al. (NIPS 2015)

\[ X = H^{(0)} \]

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

$z_n$ is the output for node $n$. The graph is represented by $z$. The adjacency matrix is denoted by $\hat{A}$.
One fits all: Classification and link prediction with GNNs/GCNs

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

Node classification:

$$\text{softmax}(z_n)$$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$$\text{softmax}(\sum_n z_n)$$

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

Link prediction:

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

Kipf & Welling (NIPS BDL 2016)

“Graph Auto-Encoders”
What do learned representations look like?

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

\[
f(\cdot) = \text{Forward pass through untrained 3-layer GCN model}
\]

Parameters initialized randomly

[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:

\[ \mathcal{L} = - \sum_{l \in \mathcal{V}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf} \]

- \( \mathcal{V}_L \) set of labeled node indices
- \( Y \) label matrix
- \( Z \) 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 \[ Z = f(X, A) = \text{softmax} \left( \hat{A} \text{ ReLU} \left( \hat{A} X W^{(0)} \right) W^{(1)} \right) \]

(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 \[ Z = f(X, A) = \text{softmax} \left( \hat{A} \text{ReLU} \left( \hat{A} X W^{(0)} \right) W^{(1)} \right) \]

**Classification results (accuracy)**

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

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

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 \(^2\)\(^3\)\(^4\)

To be presented at ICML 2018!
Motivation: Learning physical dynamics

Need to model interactions and their effect on dynamics

Simple example:

- 5 particles + their trajectories $x^t_i$
- Concatenate feature vectors
  \[ x^t = [x^t_1, x^t_2, \ldots, x^t_5] \]
- Feed into neural net
  \[ x^{t+1} = MLP(x^t) \text{ (or RNN)} \]
Motivation: Learning physical dynamics

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

**Simple example:**

- 5 particles + their trajectories $x_i^t$
- Concatenate feature vectors
  
  $x^t = [x_1^t, x_2^t, ..., x_5^t]$
- Feed into neural net
  
  $x^{t+1} = \text{MLP}(x^t)$ (or RNN)
- Done?
Motivation: Learning physical dynamics

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

**Simple example:**

- 5 particles + their trajectories $x^t_i$
- Concatenate feature vectors
  \[ x^t = [x^t_1, x^t_2, \ldots, x^t_5] \]
- Feed into neural net
  \[ x^{t+1} = \text{MLP}(x^t) \text{ (or RNN)} \]
- Done?

Works (somewhat), but we can do a lot better.
A naïve model of Intuitive Physics

Problems:
A naïve model of Intuitive Physics

Problems:

• Arbitrary ordering of nodes
  ➞ Need permutation equivariance
A naïve model of Intuitive Physics

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

Problems:

• Arbitrary ordering of nodes
  ➡ Need permutation equivariance

• Model doesn’t know about structure of interactions
  ➡ For many fundamental physical systems, interactions are pairwise

Structured neural models to the rescue!
A naïve model of Intuitive Physics

Problems:

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

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

Battaglia et al., (NIPS 2016)
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**
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Springs</th>
<th>Charged</th>
<th>Kuramoto</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5 objects</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corr. (path)</td>
<td>52.4</td>
<td>55.8</td>
<td>62.8</td>
</tr>
<tr>
<td>Corr. (LSTM)</td>
<td>55.0</td>
<td>61.8</td>
<td>55.9</td>
</tr>
<tr>
<td>NRI (sim.)</td>
<td>99.9</td>
<td>59.1</td>
<td>–</td>
</tr>
<tr>
<td>NRI (learned)</td>
<td>99.8</td>
<td>82.6</td>
<td>96.0</td>
</tr>
<tr>
<td><strong>10 objects</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corr. (path)</td>
<td>50.4</td>
<td>51.4</td>
<td>59.3</td>
</tr>
<tr>
<td>Corr. (LSTM)</td>
<td>51.0</td>
<td>52.9</td>
<td>53.9</td>
</tr>
<tr>
<td>NRI (sim.)</td>
<td>98.2</td>
<td>53.9</td>
<td>–</td>
</tr>
<tr>
<td>NRI (learned)</td>
<td>98.4</td>
<td>71.3</td>
<td>76.2</td>
</tr>
<tr>
<td>Supervised</td>
<td>98.7</td>
<td>94.9</td>
<td>96.9</td>
</tr>
</tbody>
</table>
Learning latent interaction graphs

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Springs</th>
<th>Charged</th>
<th>Kuramoto</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 objects</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corr. (path)</td>
<td>52.4</td>
<td>55.8</td>
<td>62.8</td>
</tr>
<tr>
<td>Corr. (LSTM)</td>
<td>55.0</td>
<td>61.8</td>
<td>55.9</td>
</tr>
<tr>
<td>NRI (sim.)</td>
<td>99.9</td>
<td>59.1</td>
<td>–</td>
</tr>
<tr>
<td>NRI (learned)</td>
<td>99.8</td>
<td>82.6</td>
<td>96.0</td>
</tr>
<tr>
<td>Supervised</td>
<td>99.9</td>
<td>96.3</td>
<td>99.8</td>
</tr>
</tbody>
</table>

NRI can learn to discover ground-truth relations with very high accuracy!
Learning latent interaction graphs

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)

---

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

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

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 data.

![Diagram of MolGAN](image)

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

(Under review at ICML TADGM Workshop)

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

\[ p(A_{ij}) = f(z_i, z_j) \]

e.g. \[ p(A_{ij}) = \sigma(z_i^T z_j) \]
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

\[ p(A_{ij}) = f(z_i, z_j) \]

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

\[ p(A_{ij}) = \sigma(z_i^T z_j) \]
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

\[ p(A_{ij}) = f(z_i, z_j) \]

\[ \text{e.g. } p(A_{ij}) = \sigma(z_i^T z_j) \]

**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:**

1. $h_1$
   - $S_2$
   - $1$
2. $h_2$
   - $S_3$
   - $0$
3. $h_3$
   - $S_4$
   - $1$
4. $h_4$
   - $S_5$
   - $1$
5. $h_5$
   - $S_6$
   - $1$

GraphRNN

Or in a single step:

$p(z)$

$KL$ $\tilde{A}$ $k$

$\sim$

$p_\theta(G|z)$ $\tilde{E}$ $\widetilde{F}$

GraphVAE
Likelihood-based (deep) graph generation

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

Sequentially:

```
       h1  h2  h3  h4  h5
        1   2   3   4   5
       1   0  0   0  1
S2   S3   S4   S4   S5
```

**GraphRNN**

```
Sequentially:
Sample + Edge-level RNN
Graph-level RNN
```

Or in a single step:

**GraphVAE**

```
p(z)
```

**GraphRNN**

```
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)

- **Sequentially:**
  - $h_1$ → $h_2$ → $h_3$ → $h_4$ → $h_5$
  - $1 \rightarrow 1 \rightarrow 1 \rightarrow 1 \rightarrow 1$
  - Learning Graphical State Transitions
  - Johnson (ICLR 2017)

- **GraphRNN**
  - Sample + Edge-level RNN
  - Graph-level RNN

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

- **GraphVAE**
  - Simonovsky et al.
  - (arXiv 2018)

- **GraphRNN**
  - You et al.
  - (ICML 2018)

**Or in a single step:**

- **GraphVAE**

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)

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)

\[
\begin{align*}
\mathbf{h}^{(\ell+1)}_i &= f_s(\mathbf{h}^{(\ell)}_i, \mathbf{x}_i) + \sum_{j=1}^{N} \sum_{y=1}^{Y} \frac{A_{ijy}}{|\mathcal{N}_i|} f_y(\mathbf{h}^{(\ell)}_j, \mathbf{x}_i) \\
\mathbf{h}^{(\ell+1)}_i &= \tanh(\mathbf{h}^{(\ell+1)}_i)
\end{align*}
\]
MolGAN model architecture

**Generator**: MLP to predict graph at once

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

\[
\begin{align*}
    h_{i}^{(\ell+1)} &= f_s(h_{i}^{(\ell)}, x_{i}) + \sum_{j=1}^{N} \sum_{y=1}^{Y} \frac{A_{ijy}}{|N_i|} f_y(h_{j}^{(\ell)}, x_{i}) \\
    h_{i}^{(\ell+1)} &= \tanh(h_{i}^{(\ell+1)}) ,
\end{align*}
\]

+ gated global pooling

Li et al., (ICLR 2016)
## Chemical synthesis results

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

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

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

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

**Significantly faster training/generation**
Chemical synthesis results

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

*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

![Relational reasoning diagram](image)

[Santoro et al., NIPS 2017]

Multi-Agent RL

![Multi-Agent RL diagram](image)

[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!

Source pin

[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 billion edge graph!

[Leskovec lab, Stanford]

Finding bugs in code

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

[Allamanis et al., ICLR 2018]
Conclusions

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

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

GCN for recommendation on 16 billion edge graph!

[Leskovec lab, Stanford]

Finding bugs in code

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

[Sukhbaatar et al., NIPS 2016]
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.


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