Differential Deep Learning on Graphs and its Applications

Chengxi Zang and Fei Wang
Weill Cornell Medicine

www.calvinzang.com
This Tutorial

- Friday, February 7, 2020, 2:00 PM - 6:00 PM
- Sutton North, Hilton New York Midtown, NYC
This Tutorial

- **Molecular Graph Generation**: to generate novel molecules with optimized properties
  - Graph generation
  - Graph property prediction
  - Graph optimization

- **Learning Dynamics on Graphs**: to predict temporal change or final states of complex systems
  - Continuous-time dynamics prediction
  - Structured sequence prediction
  - Node classification/regression

- **Mechanism Discovery**: to find dynamical laws of complex systems
  - Density Estimation vs. Mechanism Discovery
  - Data-driven discovery of differential equations
Part 2: Neural Dynamics on Complex Networks

Chengxi Zang and Fei Wang
Weill Cornell Medicine
www.calvinzang.com
Structures and Dynamics of Complex Systems

- Brain and Bioelectrical flow
- Transportation and Traffic flow
- Social Networks and Information flow
- Ecological Systems and Energy flow
Problem: Learning Dynamics of complex systems

Dynamics? How to predict the temporal change of these complex systems?
Learning Dynamics on Graph
- Dynamics of nodes: $X(t) \in \mathbb{R}^{n \times d}$ at $t$, where $n$ is number of nodes, $d$ is number of features, $X(t)$ changes over continuous time $t$.
- Graph: $G = (V, E)$, $V$ are nodes, $E$ are edges.
- How dynamics $\frac{dX(t)}{dt} = f(X(t), G, \theta, t)$ change on graph?
Problem: Prediction Tasks

- **Continuous-time network dynamics prediction:**
  - Input: $G, \{X(t_1), X(t_2), \ldots, X(t_T) \mid 0 \leq t_1 < \ldots < t_T\}$, $t_1 < \ldots < t_T$ are arbitrary time moments
  - Output: to predict $X(t)$ at an arbitrary time moment

(Special case) Structured sequence prediction

(Special case) Node (semi-supervised) regression/classification
Problem: Prediction Tasks

- **Continuous-time network dynamics prediction:**
  - Input: $G, \{X(t_1), X(t_2), \ldots, X(t_T)| 0 \leq t_1 < \cdots < t_T\}$, $t_1 < \cdots < t_T$ are arbitrary time moments
  - Output: to predict $X(t)$ at an arbitrary time moment

- **(Special case) Structured sequence prediction**
  - Input: $G, \{X[1], X[2], \ldots, X[T]| 0 \leq 1 < \cdots < T\}$, ordered sequence
  - Output: to predict next k steps $X[T + k]$

- **(Special case) Node (semi-supervised) regression/classification**
  - Input: $G, \hat{X} = [\hat{X}, \text{Mask} \odot \hat{Y}]$ features and node labels, only one snapshot
  - Output: to predict $[X, Y]$
Why Dynamics Matter?

- To understand, predict, and control real-world dynamic systems in engineering and science.
  - Brain dynamics, traffic dynamics, social dynamics
Challenges: Dynamics of Complex Systems

- **Complex systems:**
  - High-dimensionality and Complex interactions
  - ≥ 100 nodes, ≥ 1000 interactions

- **Dynamics:**
  - Continuous-time, Nonlinear

- **Structural-dynamic dependencies:**
  - Difficult to be modeled by simple mechanistic models
Challenges: Dynamics of Complex Systems

- **Examples of dynamics on graphs**
  - Linear Dynamics
  - Linear Dynamics
  - Non-Linear Dynamics

\[
f(X(t), G, \theta, t)
\]
Related Works 1: Learning Continuous Time Dynamics

- To learn continuous-time dynamics
  - A clear knowledge of the mechanisms, small systems, few interaction terms, first principle from physical laws, mechanistic models,

\[
\mathbf{F} = \frac{d}{dt}(m\mathbf{v})
\]

\[
\mathbf{M} = \mathbf{M} + \mathbf{C}\mathbf{M}
\]

\[
\frac{d\mathbf{x}}{dt} = \sigma(y - x),
\]

\[
\frac{dy}{dt} = x(\rho - z) - y,
\]

\[
\frac{dz}{dt} = xy + \beta z.
\]
Data-driven Dynamics for Small Systems

- Data-driven discovery of ODEs/ PDEs
  - Sparse Regression
  - Residual Network
  - Etc.

- Small systems!
  - <10 nodes & interactions
  - Combinatorial complexity
  - Not for complex systems

Image from: Brunton et al. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. PNAS
Related Works 2: Structured Sequence Learning

- **Defined characteristics**
  - Dynamics on graphs are regularly-sampled with same time intervals

- **Temporal Graph Neural Networks**
  - RNN + CNN
  - RNN + GNN
    - $X[t+1] = \text{LSTM} (\text{GCN}([t], G))$

- **Limitations:**
  - Only ordered sequence instead of continuous physical time

Seo et al. 2016. *Structured Sequence Modeling with Graph Convolutional Recurrent Networks.*
Wu et al. 2019. *A Comprehensive Survey on Graph Neural Networks*
Related Works 3: Node (Semi-supervised) Classification/Regression

- **Defined characteristics**
  - One-snapshot features and some labels on graphs
  - Goal: to assign labels to each node

- **Graph Neural Networks**
  - GCN,
  - GAT, etc.

- **Limitations**
  - 1 or 2 layers
  - Lacking a continuous-time dynamics view
    - To spread features or labels on graphs
    - Continuous-time: more fine-grained control on diffusion

Kipf et al. 2016. *Semi-Supervised Classification with Graph Convolutional Networks*
Velickovic et al. 2017. *Graph Attention Networks*
Continuous-time network dynamics prediction:
- Input: $G$, $\{X(t_1), X(t_2), \ldots, X(t_T)\} | 0 \leq t_1 < \cdots < t_T$, $t_1 < \cdots < t_T$ are arbitrary time moments
- Model: dynamics on graphs $\frac{dX(t)}{dt} = f(X(t), G, \theta, t)$
- Output: to predict $X(t)$ at an arbitrary time moment

(Special case) Structured sequence prediction
- Input: $G$, $\{X[1], X[2], \ldots, X[T]\} | 0 \leq 1 < \cdots < T$, ordered sequence
- Model: dynamics on graphs $\frac{dX(t)}{dt} = f(X(t), G, \theta, t)$
- Output: to predict next k steps $X[T + k]$

(Special case) Node (semi-supervised) regression/classification
- Input: $G$, $\hat{X} = [\hat{X}, \text{Mask} \odot \hat{Y}]$ features and node labels, only one snapshot
- Model: dynamics on graphs $\frac{dX(t)}{dt} = f(X(t), G, \theta, t)$
- Output: to predict $[X, Y]$
Our Ideas

- **Differential Equation Systems**
  - Graphs and Differential Equations are general tools to describe structures and dynamics of complex systems

- **Deep Learning**
  - RNN, GNN, Temporal GNN, Res-Net etc. are the state-of-the-art computational tools driven by data

- **How to leverage Differential equation systems and Deep Learning?**
Neural Dynamics on Complex Networks (NDCN)

- **Differential Deep Learning**
  - Differential Equation systems: \( \frac{dX(t)}{dt} = f(X(t), G, W, t) \) is a graph neural network like structure.
  - Differential Deep model: \( X(t) = X(0) + \int_0^t f(X(\tau), G, W, \tau) d\tau \) for arbitrary time \( t \)
  - Learned as following optimization problem:

\[
\begin{align*}
\text{argmin}_{W_*, b_*} & \quad \mathcal{L} = \int_0^T |X(t) - \hat{X}(t)| \, dt \\
\text{subject to} & \quad X_h(t) = \tanh \left( X(t)W_e + b_e \right) W_0 + b_0 \\
& \quad \frac{dX_h(t)}{dt} = \text{ReLU} \left( \Phi X_h(t)W + b \right), X_h(0) \\
& \quad X(t) = X_h(t)W_d + b_d
\end{align*}
\]

\( \Phi = D^{-\frac{1}{2}} (D - A) D^{-\frac{1}{2}} \in \mathbb{R}^{n \times n} \)
Neural Dynamics on Complex Networks (NDCN)

- **Differential Deep Learning**
  - Differential Equation systems: \( \frac{dX(t)}{dt} = f(X(t), G, W, t) \)
    - is a graph neural network like structure.
  - Differential Deep model: \( X(t) = X(0) + \int_0^t f(X(\tau), G, W, \tau) d\tau \) for arbitrary time \( t \)
  - Learned as following optimization problem:

\[
\arg\min_{W^*, b^*} \mathcal{L} = \int_0^T |X(t) - \hat{X}(t)| dt
\]

subject to

\[
X_h(t) = \tanh \left( X(t)W_e + b_e \right) W_0 + b_0
\]

\[
\frac{dX_h(t)}{dt} = \text{ReLU} \left( \Phi X_h(t)W + b \right), X_h(0)
\]

\[
X(t) = X_h(t)W_d + b_d
\]

\[
\Phi = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \in \mathbb{R}^{n \times n}
\]
Interpretation from Residual Learning

- **Deep Learning**: $f^*$ is a neural layer
  - Each layer: $X[l + 1] = f_{l+1}(X[l])$
  - Deep Model: $X[L] = f_L \circ \cdots \circ f_1(X[0])$

- **Residual Learning**: deep
  - Each layer: $X[l + 1] = X[l] + f_{l+1}(X[l])$
  - Deep Model: $X[L] = (f_L + I) \circ \cdots \circ (f_1+I)(X[0])$

- **Differential Deep Learning**:
  - Each time moment ("layer"): Instantaneous rate at $t$: $\frac{dx}{dt} = f(X(t))$
    - Each Discrete layer vs. continuous time moment
    - Neural mapping vs. Neural Differential Equation Systems
  - Continuous-time ("Deep") Model: $X(t) = X(0) + \int_0^t f(X(\tau), W, \tau) d\tau$ Integration over continuous-time
    - A sequence of mappings vs. continuous integration
    - Trajectory of dynamics
GNN, Residual-GNN, ODE-GNN, NDCN

- GNN: $X_{t+1} = f(G, X_t, \theta_t)$
- Residual-GNN: $X_{t+1} = X_t + f(G, X_t, \theta_t)$
- Differential-GNN: $X_{t+\delta} = X_t + \delta f(G, X_t, \theta_t), \delta \to 0$
  \[ \frac{dX}{dt} = f(G, X_t, \theta_t) \]

- Our model is an Differential-GNN with continuous layer with real number depth.
Interpretation from RNN and Temporal GNN

- **RNN, Temporal GNN and our model**
  - RNN or Temporal GNN
    - \( h_t = f(h_{t-1}, x_t, \theta_t) \) or \( h_t = f(h_{t-1}, G \ast x_t, \theta_t) \)
    - \( y_t = o(h_t, w_t) \)
  - Residual RNN or Temporal GNN with skip connection
    - \( h_t = h_{t-1} + f(h_{t-1}, x_t, \theta_t) \) or \( h_t = h_{t-1} + f(h_{t-1}, G \ast x_t, \theta_t) \)
    - \( y_t = o(h_t, w_t) \)
  - Differential RNN or Differential GNN
    - \( \frac{dh_t}{dt} = f(h_t, x_t, \theta_t) \) or \( \frac{dh_t}{dt} = f(h_t, G \ast x_t, \theta_t) \)
    - \( y_t = o(h_t, w_t) \)

- **Our model is an Differential GNN**
  - Learning continuous-time network dynamics
  - Encompassing Temporal GNN by discretization
  - Encompassing RNN by not using graph convolution
Exp1: Learning Continuous-time Network Dynamics

The Problem:
- Input: \( \{X(t_1), X(t_2), \ldots, X(t_T)\} \) with \( 0 \leq t_1 < \cdots < t_T \), \( t_1 < \cdots < t_T \) are arbitrary time moments with different time intervals
- Output: \( X(t) \), \( t \) is an arbitrary time moment
  - **interpolation** prediction: \( t < t_T \) and \( \neq \{t_1 < \cdots < t_T\} \)
  - **extrapolation** prediction: \( t > t_T \)

Setups:
- 120 irregularly sampled snapshots of network dynamics
- First 100: 80 for train 20 for testing interpolation
- Last 20: testing for extrapolation
Real-world Dynamics on Graph (adjacency matrix A)

- **Heat diffusion:** \[
\frac{dx_i(t)}{dt} = -k_{i,j} \sum_{j=1}^{n} A_{i,j} (x_i(t) - x_j(t))
\]

- **Mutualistic interaction:** \[
\frac{dx_i(t)}{dt} = b_i + x_i(t) \left(1 - \frac{x_i(t)}{k_i}\right) \left(\frac{x_i(t)}{c_i} - 1\right) + \sum_{j=1}^{n} A_{i,j} \frac{x_i(t) \times x_j(t)}{d_i + e_i x_i(t) + h_i x_j(t)}
\]

- **Gene regulatory:** \[
\frac{dx_i(t)}{dt} = -b_i x_i(t)^f + \sum_{j=1}^{n} A_{i,j} \frac{x_j(t)^h}{x_j(t)^{h+1}}
\]

Graphs

- Grid, Random, power-law, small-world, community, etc.

Visualizing dynamics on graph

- Nodes are numbered by community labels
- Mapped into a $\mathbb{N}^2$ grid
- $X(t)^{n \times 1}: \mathbb{N}^2 \rightarrow \mathbb{R}$
Exp1: Learning Continuous-time Network Dynamics

- Baselines: ablation models
  - Differential-GNN
    - No encoding layer
  - Neural ODE Network
    - No graph diffusion
  - NDCN without control parameter $W$
    - Determined dynamics

Chen et al. 2019. Neural Ordinary Differential Equations, NeurIPS.
Exp1: Heat Diffusion on Different Graphs
Exp1: Mutualistic Dynamics on Different Graphs
Exp1: Gene Dynamics on Different Graphs
Exp1: Results for Continuous-time Extrapolation

- Mean Absolute Percentage Error
- 20 runs for 3 dynamics on 5 graphs
- Our model achieves lowest error

Table 1: Continuous-time Extrapolation Prediction. Our NDCN predicts different continuous-time network dynamics accurately. Each result is the normalized $\ell_1$ error with standard deviation (in percentage %) from 20 runs for 3 dynamics on 5 networks by each method.

<table>
<thead>
<tr>
<th></th>
<th>Grid</th>
<th>Random</th>
<th>Power Law</th>
<th>Small World</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heat Diffusion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No-Encode</td>
<td>29.9 ± 7.3</td>
<td>27.8 ± 5.1</td>
<td>24.9 ± 5.2</td>
<td>24.8 ± 3.2</td>
<td>30.2 ± 4.4</td>
</tr>
<tr>
<td>No-Graph</td>
<td>30.5 ± 1.7</td>
<td>5.8 ± 1.3</td>
<td>6.8 ± 0.5</td>
<td>10.7 ± 0.6</td>
<td>24.3 ± 3.0</td>
</tr>
<tr>
<td>No-Control</td>
<td>73.4 ± 14.4</td>
<td>28.2 ± 4.0</td>
<td>25.2 ± 4.3</td>
<td>30.8 ± 4.7</td>
<td>37.1 ± 3.7</td>
</tr>
<tr>
<td>NDCN</td>
<td>4.1 ± 1.2</td>
<td>4.3 ± 1.6</td>
<td>4.9 ± 0.5</td>
<td>2.5 ± 0.4</td>
<td>4.8 ± 1.0</td>
</tr>
<tr>
<td>Mutualistic Interaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No-Encode</td>
<td>45.3 ± 3.7</td>
<td>9.1 ± 2.9</td>
<td>29.9 ± 8.8</td>
<td>54.5 ± 3.6</td>
<td>14.5 ± 5.0</td>
</tr>
<tr>
<td>No-Graph</td>
<td>56.4 ± 1.1</td>
<td>6.7 ± 2.8</td>
<td>14.8 ± 6.3</td>
<td>54.5 ± 1.0</td>
<td>9.5 ± 1.5</td>
</tr>
<tr>
<td>No-Control</td>
<td>140.7 ± 13.0</td>
<td>10.8 ± 4.3</td>
<td>106.2 ± 42.6</td>
<td>115.8 ± 12.9</td>
<td>16.9 ± 3.1</td>
</tr>
<tr>
<td>NDCN</td>
<td>26.7 ± 4.7</td>
<td>3.8 ± 1.8</td>
<td>7.4 ± 2.6</td>
<td>14.4 ± 3.3</td>
<td>3.6 ± 1.5</td>
</tr>
<tr>
<td>Gene Regulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No-Encode</td>
<td>31.7 ± 14.1</td>
<td>17.5 ± 13.0</td>
<td>33.7 ± 9.9</td>
<td>25.5 ± 7.0</td>
<td>26.3 ± 10.4</td>
</tr>
<tr>
<td>No-Graph</td>
<td>13.3 ± 0.9</td>
<td>12.2 ± 0.2</td>
<td>43.7 ± 0.3</td>
<td>15.4 ± 0.3</td>
<td>19.6 ± 0.5</td>
</tr>
<tr>
<td>No-Control</td>
<td>65.2 ± 14.2</td>
<td>68.2 ± 6.6</td>
<td>70.3 ± 7.7</td>
<td>58.6 ± 17.4</td>
<td>64.2 ± 7.0</td>
</tr>
<tr>
<td>NDCN</td>
<td>16.0 ± 7.2</td>
<td>1.8 ± 0.5</td>
<td>3.6 ± 0.9</td>
<td>4.3 ± 0.9</td>
<td>2.5 ± 0.6</td>
</tr>
</tbody>
</table>
Exp1: Results for Continuous-time Interpolation

- Interpolation is easier than extrapolation
- Our model achieves lowest error

Table 2: Continuous-time Interpolation Prediction. Our NDCN predicts different continuous-time network dynamics accurately. Each result is the normalized $\ell_1$ error with standard deviation (in percentage %) from 20 runs for 3 dynamics on 5 networks by each method.

<table>
<thead>
<tr>
<th></th>
<th>Grid</th>
<th>Random</th>
<th>Power Law</th>
<th>Small World</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heat Diffusion</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No-Encode</td>
<td>32.0 ± 12.7</td>
<td>26.7 ± 4.4</td>
<td>25.7 ± 3.8</td>
<td>27.9 ± 7.3</td>
<td>35.0 ± 6.3</td>
</tr>
<tr>
<td>No-Graph</td>
<td>41.9 ± 1.8</td>
<td>9.4 ± 0.6</td>
<td>18.2 ± 1.5</td>
<td>25.0 ± 2.1</td>
<td>25.0 ± 1.4</td>
</tr>
<tr>
<td>No-Control</td>
<td>56.8 ± 2.8</td>
<td>32.2 ± 7.0</td>
<td>33.5 ± 5.7</td>
<td>40.4 ± 3.4</td>
<td>39.1 ± 4.5</td>
</tr>
<tr>
<td><strong>NDCN</strong></td>
<td><strong>3.2 ± 0.6</strong></td>
<td><strong>3.2 ± 0.4</strong></td>
<td><strong>5.6 ± 0.6</strong></td>
<td><strong>3.4 ± 0.4</strong></td>
<td><strong>4.3 ± 0.5</strong></td>
</tr>
<tr>
<td><strong>Mutualistic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No-Encode</td>
<td>28.9 ± 2.0</td>
<td>19.9 ± 6.5</td>
<td>34.5 ± 13.4</td>
<td>27.6 ± 2.6</td>
<td>25.5 ± 8.7</td>
</tr>
<tr>
<td>No-Graph</td>
<td>28.7 ± 4.5</td>
<td>7.8 ± 2.4</td>
<td>23.2 ± 4.2</td>
<td>26.9 ± 3.8</td>
<td>14.1 ± 2.4</td>
</tr>
<tr>
<td>No-Control</td>
<td>72.2 ± 4.1</td>
<td>22.5 ± 10.2</td>
<td>63.8 ± 3.9</td>
<td>67.9 ± 2.9</td>
<td>33.9 ± 12.3</td>
</tr>
<tr>
<td><strong>NDCN</strong></td>
<td><strong>7.6 ± 1.1</strong></td>
<td><strong>6.6 ± 2.4</strong></td>
<td><strong>6.5 ± 1.3</strong></td>
<td><strong>4.7 ± 0.7</strong></td>
<td><strong>7.9 ± 2.9</strong></td>
</tr>
<tr>
<td><strong>Gene Regulation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No-Encode</td>
<td>39.2 ± 13.0</td>
<td>14.5 ± 12.4</td>
<td>33.6 ± 10.1</td>
<td>27.7 ± 9.4</td>
<td>21.2 ± 10.4</td>
</tr>
<tr>
<td>No-Graph</td>
<td>25.2 ± 2.3</td>
<td>11.9 ± 0.2</td>
<td>39.4 ± 1.3</td>
<td>15.7 ± 0.7</td>
<td>18.9 ± 0.3</td>
</tr>
<tr>
<td>No-Control</td>
<td>66.9 ± 8.8</td>
<td>31.7 ± 5.2</td>
<td>40.3 ± 6.6</td>
<td>49.0 ± 8.0</td>
<td>35.5 ± 5.3</td>
</tr>
<tr>
<td><strong>NDCN</strong></td>
<td><strong>5.8 ± 1.0</strong></td>
<td><strong>1.5 ± 0.6</strong></td>
<td><strong>2.9 ± 0.5</strong></td>
<td><strong>4.2 ± 0.9</strong></td>
<td><strong>2.3 ± 0.6</strong></td>
</tr>
</tbody>
</table>
Exp2: Structured Sequence Prediction

- **The Problem (Structured sequence prediction):**
  - Input: \( \{X[1], X[2], \ldots, X[T] \mid 0 \leq 1 < \ldots < T \} \), \( 1, \ldots T \) are regularly-sampled with same time intervals
    - with an emphasis on ordered sequence rather than time
  - Output: \( X(t_T + M) \), next \( M \) steps
    - extrapolation prediction

- **Setups:**
  - 100 regularly sampled snapshots of network dynamics
  - First 80 for training, last 20 for testing
Exp2: Structured Sequence Prediction

- Baselines: temporal-GNN models
  - LSTM-GNN
    - $X[t+1] = \text{LSTM}(\text{GCN}([t], G))$
  - GRU-GNN
    - $X[t+1] = \text{GRU}(\text{GCN}([t], G))$
  - RNN-GNN
    - $X[t+1] = \text{RNN}(\text{GCN}([t], G))$

Seo et al. 2016. Structured Sequence Modeling with Graph Convolutional Recurrent Networks.
Wu et al. 2019. A Comprehensive Survey on Graph Neural Networks
Exp2: Structured Sequence Prediction

- **Results:**
  - Our model achieves lowest error with much less parameters

- **The learnable parameters:**
  - LSTM-GNN: 84,890, GRU-GNN: 64,770, RNN-GNN: 24,530
  - NDCN: 901

Table 3: Regularly-sampled Extrapolation Prediction. Our NDCN predicts different structured sequences accurately. Each result is the normalized $\ell_1$ error with standard deviation (in percentage %) from 20 runs for 3 dynamics on 5 networks by each method.

<table>
<thead>
<tr>
<th></th>
<th>Grid</th>
<th>Random</th>
<th>Power Law</th>
<th>Small World</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heat Diffusion</strong></td>
<td>LSTM-GNN</td>
<td>12.8 ± 2.1</td>
<td>21.6 ± 7.7</td>
<td>12.4 ± 5.1</td>
<td>11.6 ± 2.2</td>
</tr>
<tr>
<td></td>
<td>GRU-GNN</td>
<td>11.2 ± 2.2</td>
<td>9.1 ± 2.3</td>
<td>8.8 ± 1.3</td>
<td>9.3 ± 1.7</td>
</tr>
<tr>
<td></td>
<td>RNN-GNN</td>
<td>18.8 ± 5.9</td>
<td>25.0 ± 5.6</td>
<td>18.9 ± 6.5</td>
<td>21.8 ± 3.8</td>
</tr>
<tr>
<td></td>
<td>NDCN</td>
<td>4.3 ± 0.7</td>
<td>4.7 ± 1.7</td>
<td>5.4 ± 0.4</td>
<td>2.7 ± 0.4</td>
</tr>
<tr>
<td><strong>Mutualistic Interaction</strong></td>
<td>LSTM-GNN</td>
<td>51.4 ± 3.3</td>
<td>24.2 ± 24.2</td>
<td>27.0 ± 7.1</td>
<td>58.2 ± 2.4</td>
</tr>
<tr>
<td></td>
<td>GRU-GNN</td>
<td>49.8 ± 4.1</td>
<td>1.0 ± 3.6</td>
<td>12.2 ± 0.8</td>
<td>51.1 ± 4.7</td>
</tr>
<tr>
<td></td>
<td>RNN-GNN</td>
<td>56.6 ± 0.1</td>
<td>8.4 ± 11.3</td>
<td>12.0 ± 0.4</td>
<td>57.4 ± 1.9</td>
</tr>
<tr>
<td></td>
<td>NDCN</td>
<td>29.8 ± 1.6</td>
<td>4.7 ± 1.1</td>
<td>11.2 ± 5.0</td>
<td>15.9 ± 2.2</td>
</tr>
<tr>
<td><strong>Gene Regulation</strong></td>
<td>LSTM-GNN</td>
<td>27.7 ± 3.2</td>
<td>67.3 ± 14.2</td>
<td>38.8 ± 12.7</td>
<td>13.1 ± 2.0</td>
</tr>
<tr>
<td></td>
<td>GRU-GNN</td>
<td>24.2 ± 2.8</td>
<td>50.9 ± 6.4</td>
<td>35.1 ± 15.1</td>
<td>11.1 ± 1.8</td>
</tr>
<tr>
<td></td>
<td>RNN-GNN</td>
<td>28.0 ± 6.8</td>
<td>56.5 ± 5.7</td>
<td>42.0 ± 12.8</td>
<td>14.0 ± 5.3</td>
</tr>
<tr>
<td></td>
<td>NDCN</td>
<td>18.6 ± 9.9</td>
<td>2.4 ± 0.9</td>
<td>4.1 ± 1.4</td>
<td>5.5 ± 0.8</td>
</tr>
</tbody>
</table>
The Problem:
- One-snapshot case
- Input: G, X, part of labels Y(X)
- Output: To Complete Y(X)

Datasets:
- Table 11: Statistics for three real-world citation network datasets. N, E, D, C represent number of nodes, edges, features, classes respectively.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>N</th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>Train/Valid/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>5,429</td>
<td>1,433</td>
<td>7</td>
<td>140/500/1,000</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,327</td>
<td>4,732</td>
<td>3,703</td>
<td>6</td>
<td>120/500/1,000</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19,717</td>
<td>44,338</td>
<td>500</td>
<td>3</td>
<td>60/500/1,000</td>
</tr>
</tbody>
</table>
Exp3. Node Semi-supervised Classification

- **Baselines**
  - Graph Convolution Network (GCN)
  - Attention-based GNN (AGNN)
  - Graph Attention Networks (GAT)

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

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

Kipf et al. 2016. *Semi-Supervised Classification with Graph Convolutional Networks*

Velickovic et al. 2017. *Graph Attention Networks*
Exp3. Node Semi-supervised Classification

- **Interpretation of model**
  - Input: \( G, [X, \text{Mask} \odot Y] \), features and some node labels
  - Output: To Complete \( Y \)
  - Model: A graph dynamics to spread features and labels over time \( T \)

\[
\frac{d[X,Y]}{dt} = f(G, X, Y, W)
\]

\[
\begin{align*}
\arg\min_{w_e, b_e, w_d, b_d} \quad & \mathcal{L} = \int_0^T \mathcal{R}(t) \, dt - \sum_{i=1}^n \sum_{k=1}^c \hat{Y}_{i,k}(T) \log Y_{i,k}(T) \\
\text{subject to} \quad & X_h(0) = \tanh \left( X(0)W_e + b_e \right) \\
& \frac{dX_h(t)}{dt} = \text{ReLU} \left( \Phi X_h(t) \right) \\
& Y(T) = \text{softmax}(X_h(T)W_d + b_d)
\end{align*}
\]
Exp3. Node Semi-supervised Classification

- **Metrics**
  - Accuracy over 100 runs

- **Results**
  - Continuous-time dynamics on graphs
  - Best results at time $T=1.2$
    - Continuous depth/time
  - Not using dropout

---

**Table 4:** Test mean accuracy with standard deviation in percentage (%) over 100 runs. Our NDCN model gives very competitive results compared with many GNN models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td>AGNN</td>
<td>$83.1 \pm 0.1$</td>
<td>$71.7 \pm 0.1$</td>
<td>$79.9 \pm 0.1$</td>
</tr>
<tr>
<td>GAT</td>
<td>$83.0 \pm 0.7$</td>
<td>$72.5 \pm 0.7$</td>
<td>$79.0 \pm 0.3$</td>
</tr>
<tr>
<td>NDCN</td>
<td>$83.3 \pm 0.6$</td>
<td>$73.1 \pm 0.6$</td>
<td>$79.8 \pm 0.4$</td>
</tr>
</tbody>
</table>

---

**Figure 5:** Our NDCN model captures continuous-time dynamics. Mean classification accuracy of 100 runs over terminal time when given a specific $\alpha$. Insets are the accuracy over the two-dimensional space of terminal time and $\alpha$. 
Summary

- **Our NDCN, a unified framework to solve**
  - Continuous-time network dynamics prediction:
    - Structured sequence prediction
    - Node regression/classification at final state
gen: good performance with less parameters.

- **Differential Deep Learning on Graphs**
  - A potential data-driven method to model structure and dynamics of complex systems in a unified framework
This Tutorial

- **Molecular Graph Generation**: to generate novel molecules with optimized properties
  - Graph generation
  - Graph property prediction
  - Graph optimization

- **Learning Dynamics on Graphs**: to predict temporal change or final states of complex systems
  - Continuous-time dynamics prediction
  - Structured sequence prediction
  - Node classification/regression

- **Mechanism Discovery**: to find dynamical laws of complex systems
  - Density Estimation vs. Mechanism Discovery
  - Data-driven discovery of differential equations
This Tutorial

- AAAI-2020
- Friday, February 7, 2020, 2:00 PM - 6:00 PM
- Sutton North, Hilton New York Midtown, NYC
Differential Deep Learning on Graphs and its Applications

Chengxi Zang and Fei Wang
Weill Cornell Medicine
www.calvinzang.com