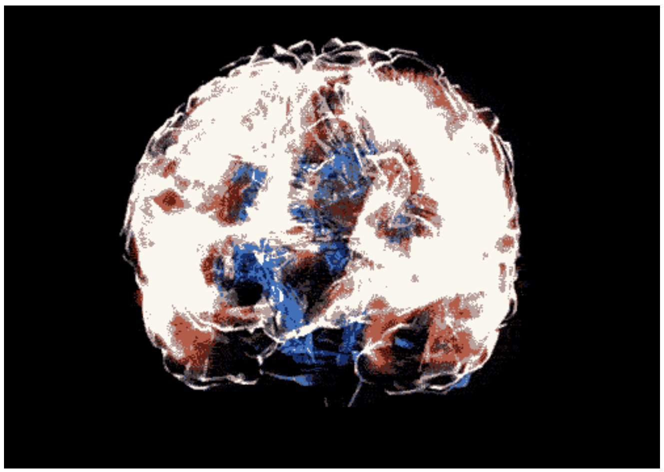
Neural Dynamics on Complex Networks

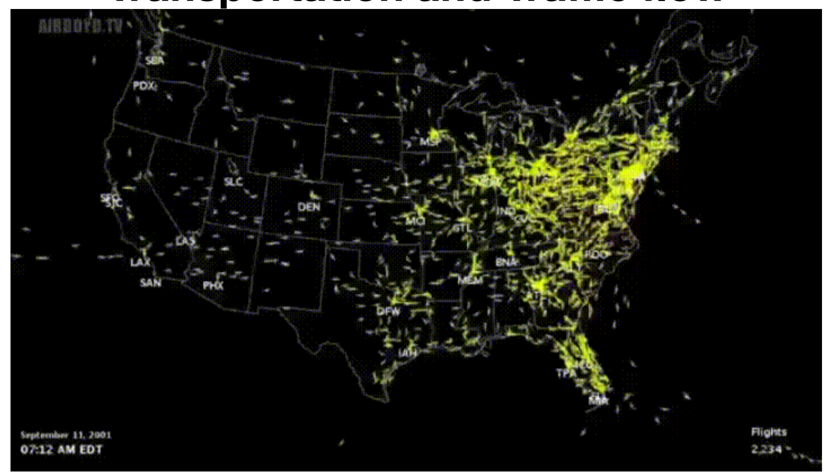
Chengxi Zang and Fei Wang
Weill Cornell Medicine
www.calvinzang.com

Structures and Dynamics of Complex Systems

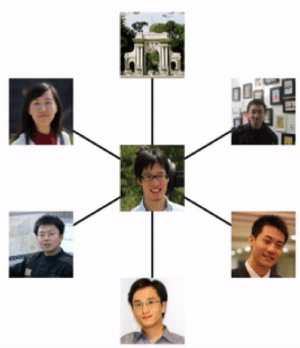
Brain and Bioelectrical flow



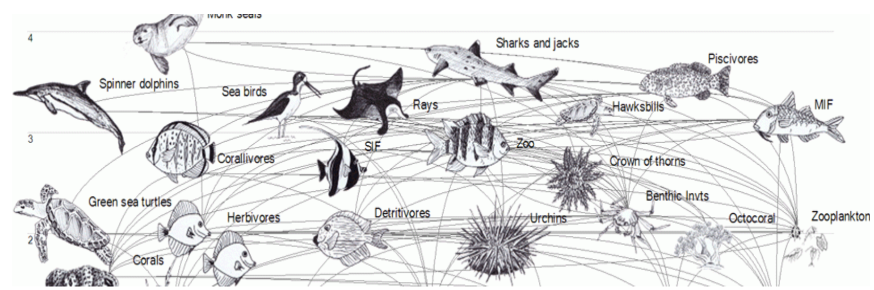
Transportation and Traffic flow



Social Networks and Information flow

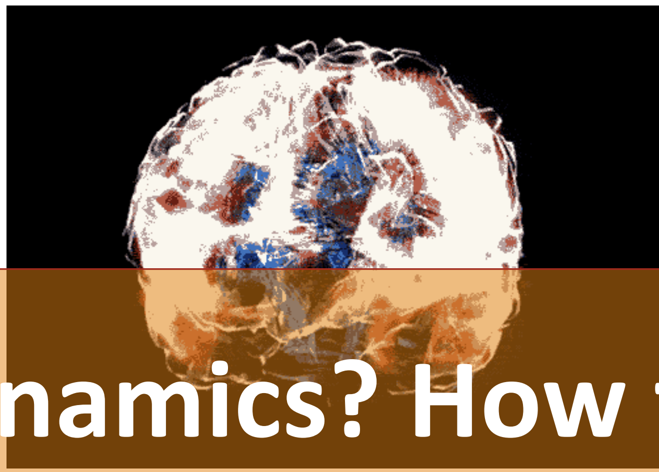


Ecological Systems and Energy flow



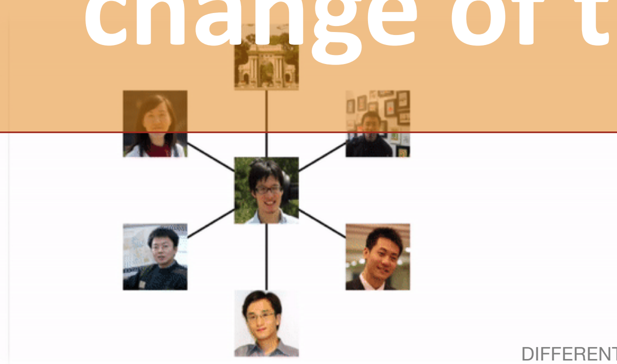
Problem: Learning Dynamics of complex systems

Brain and Bioelectrical flow

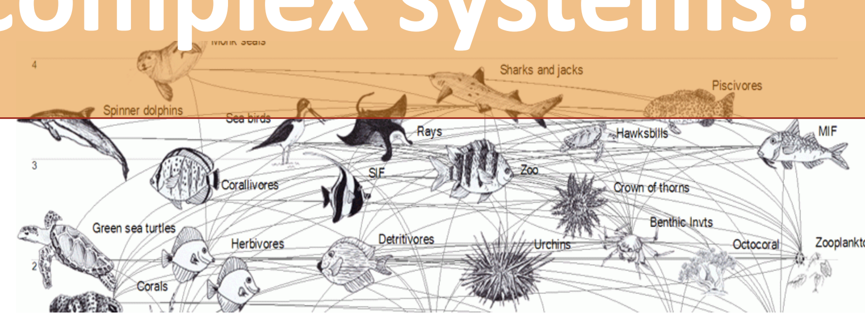


Transportation and Traffic flow

Social Networks and Information flow



Ecological Systems and Energy flow



Dynamics? How to predict the temporal change of these complex systems?

Problem: Math Formulation

❑ Learning Dynamics on Graph

- Dynamics of nodes: $X(t) \in \mathbb{R}^{n*d}$ 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?

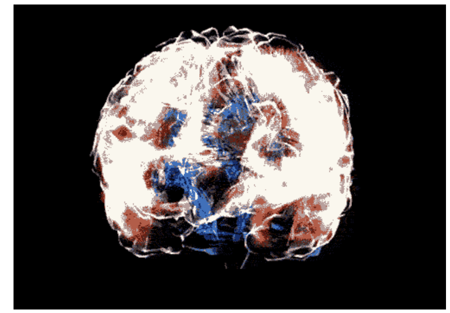
Problem: Prediction Tasks

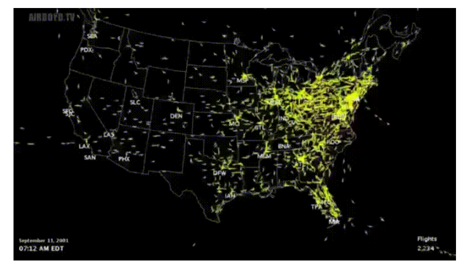
- **Continuous-time network dynamics prediction:**
  - Input: G, $\left\{\widehat{X(t_1)}, \widehat{X(t_2)}, \ldots, \widehat{X(t_T)} \middle| 0 \le t_1 < \cdots < t_T\right\}, t_1 < \cdots < t_T$ are arbitrary time moments
  - ?A model of dynamics on graphs $\frac{dX(t)}{dt} = f(X(t), G, \theta, t)$
  - Output: to predict $X(t)$ at an arbitrary time moment

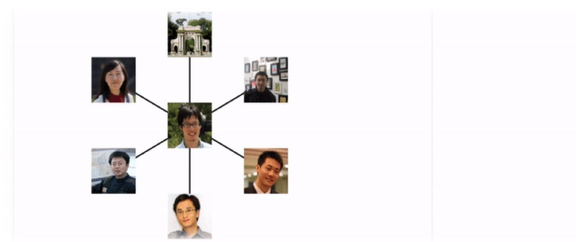
Problem: Prediction Tasks

- **Continuous-time network dynamics prediction:**
  - Input: G, $\left\{\widehat{X(t_1)}, \widehat{X(t_2)}, \dots, \widehat{X(t_T)} \middle| 0 \le t_1 < \cdots < t_T\right\}, t_1 < \cdots < t_T$ are arbitrary time moments
  - ?A model of 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, $\left\{\widehat{X[1]}, \widehat{X[2]}, \dots, \widehat{X[T]} \middle| 0 \le 1 < \cdots < T\right\}$, ordered sequence
  - ? A model of 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, $\hat{X} = [\hat{X}, Mask \odot \hat{Y}]$ features and node labels, only one snapshot
  - ? A model of dynamics on graphs $\frac{dX(t)}{dt} = f(X(t), G, \theta, t)$
  - Output: to predict $[X, Y]$

Challenges: Dynamics of Complex Systems



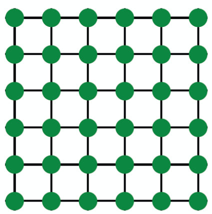




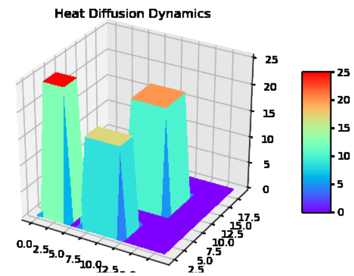
- **Complex systems:**
  - High-dimensionality and Complex interactions
  - ≥ 100 nodes, ≥ 1000 interactions
- **Dynamics:**
  - Continuous-time, Nonlinear
- **Structural-dynamic dependencies**:
  - Difficult to be modeled by simple mechanistic models

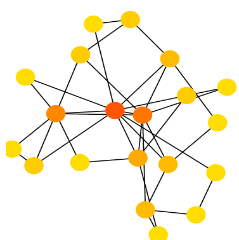
Challenges: Dynamics of Complex Systems

❑ Examples of dynamics on graphs

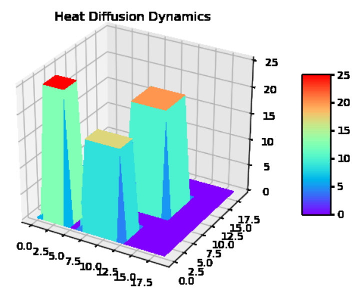


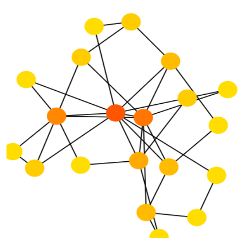
+ Linear Dynamics →



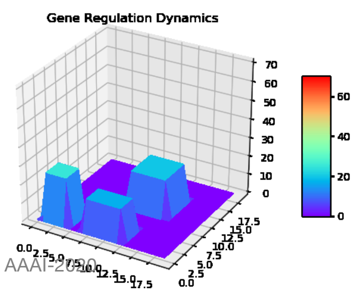


+ Linear Dynamics →





+ Non-Linear Dynamics →



$$\boldsymbol{f(X(t), G, \theta, t)}$$

?

Related Works 1: Learning Continuous Time Dynamics

- ❑ **Data-driven discovery of ODEs/ PDEs**
  - ○ **Sparse Regression**
  - ○ Residual Network
  - ○ Etc.
- ❑ **Small systems!**
  - ○ <10 nodes & interactions
  - ○ Combinatorial complexity
  - ○ Not for complex systems

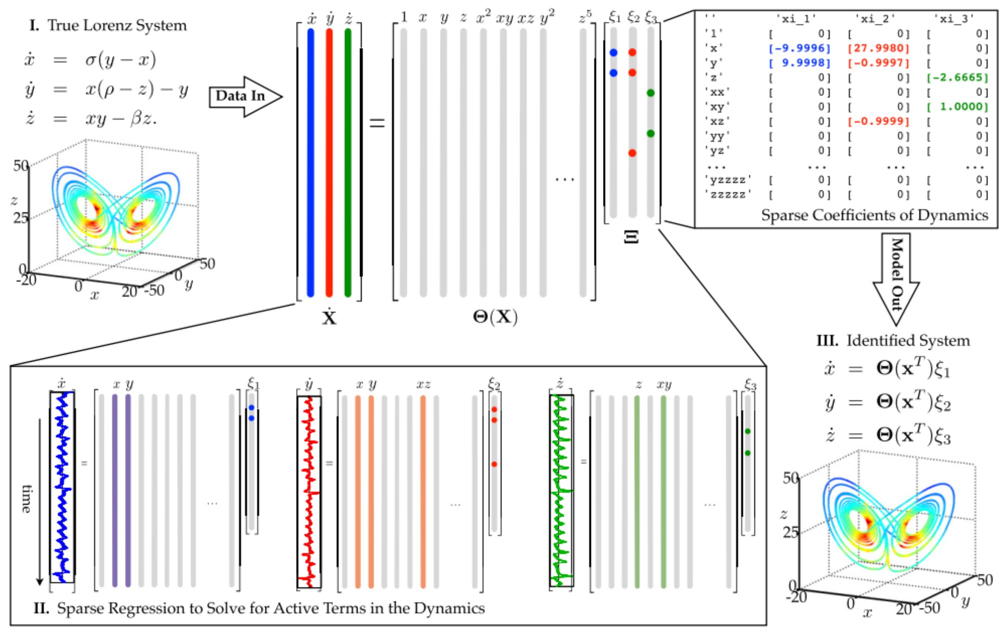


Image from: Brunton et al. 2016. [Discovering governing equations from data by sparse identification of nonlinear dynamical systems.](#) PNAS

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]=LSTM(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.

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

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

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

Neural Dynamics on Complex Networks (NDCN)

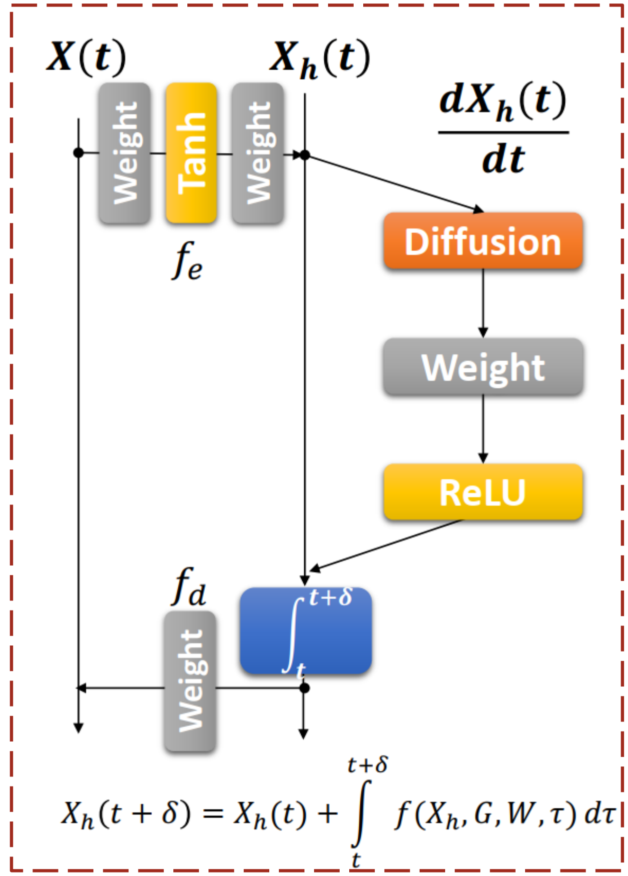
- **Differential Deep Learning**
  - Differential Equation systems: $\frac{dX(t)}{dt} = f(X(\mathrm{t}), \mathrm{G}, W, \mathrm{t})$ is a graph neural network like structure.
  - Differential Deep model: $X(t) = X(0) + \int_0^t f(X(\tau), \mathrm{G}, W, \tau) d\tau$ for arbitrary time t
  - Learned as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

$$\underset{W_*, b_*}{\operatorname{argmin}} \quad \mathcal{L} = \int_0^T |X(t) - \hat{X(t)}| \, dt$$

$$\text{subject to} \quad X_h(t) = \tanh\Big(X(t)W_e + b_e\Big)W_0 + b_0$$

$$\frac{dX_h(t)}{dt} = \mathrm{ReLU}\Big(\Phi X_h(t) W + b\Big), X_h(0)$$

$$X(t) = X_h(t) W_d + b_d$$



$\Phi = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \in \mathbb{R}^{n \times n}$

Exp1: Learning Continuous-time Network Dynamics

The Problem:

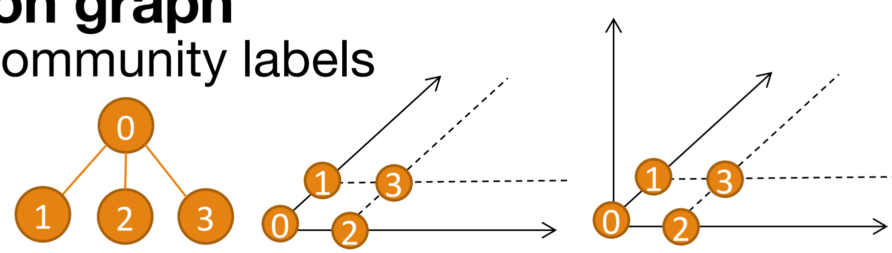
- Input: $\left\{\widehat{X(t_1)}, \widehat{X(t_2)}, \dots, \widehat{X(t_T)} \middle| 0 \le t_1 < \cdots < t_T\right\}, t_1 < \cdots < t_T$ are <u>**arbitrary time moments with different time intervals**</u>
- Output: $X(t)$, t is an arbitrary time moment
  - **interpolation** prediction: $t < t_T$ and $\neq \{t_1 < \cdots < t_T\}$
  - **extrapolation** prediction: t $> t_T$

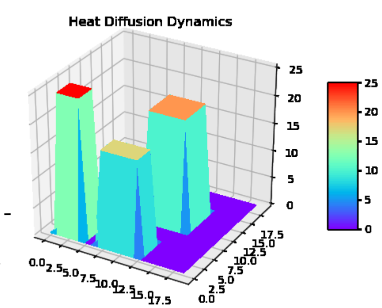
Setups:

- 120 irregularly sampled snapshots of network dynamics
- First 100: 80 for train 20 for testing interpolation
- Last 20: testing for extrapolation

Canonical Dynamics on Graphs in Physics and Biology

- **Real-world Dynamics on Graph (adjacency matrix A)**
  - **Heat diffusion**: $\frac{d\overrightarrow{x_i(t)}}{dt} = -k_{i,j}\sum_{j=1}^{n} A_{i,j}\left(\overrightarrow{x_i(t)} - \overrightarrow{x_j(t)}\right)$
  - **Mutualistic interaction**: $\frac{d\overrightarrow{x_i(t)}}{dt} = b_i + \overrightarrow{x_i(t)}\left(1 - \frac{\overrightarrow{x_i(t)}}{k_i}\right)\left(\frac{\overrightarrow{x_i(t)}}{c_i} - 1\right) + \sum_{j=1}^{n} A_{i,j}\,\frac{\overrightarrow{x_i(t)} * \overrightarrow{x_j(t)}}{d_i + e_i\overrightarrow{x_i(t)} + h_j\overrightarrow{x_j(t)}}$
  - **Gene regulatory**: $\frac{d\overrightarrow{x_i(t)}}{dt} = -b_i\overrightarrow{x_i(t)}^{f} + \sum_{j=1}^{n} A_{i,j}\,\frac{\overrightarrow{x_j(t)}^{h}}{\overrightarrow{x_j(t)}^{h} + 1}$
- **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
  - $X(t)^{n*1}$: $\mathbb{N}^2 \rightarrow \mathbb{R}$

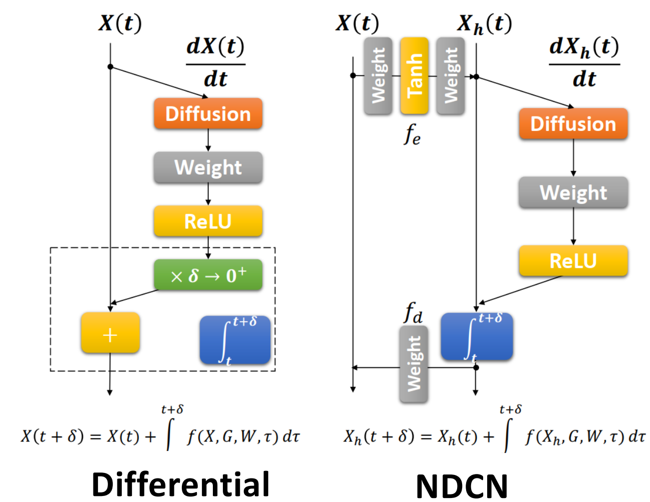




Exp1: Learning Continuous-time Network Dynamics

❑ Baselines: ablation models

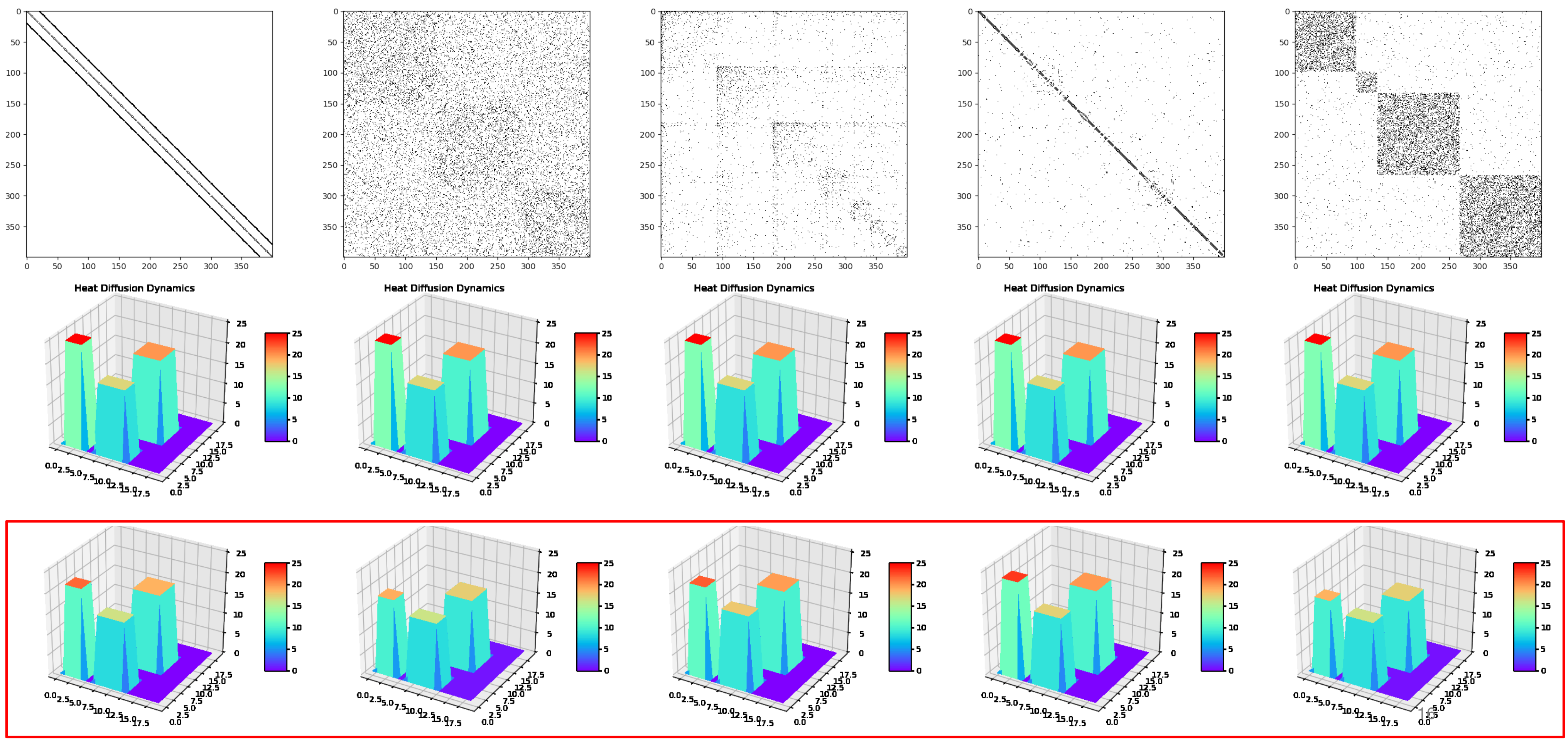
- ○ Differential-GNN
  - ❖ No encoding layer
- ○ Neural ODE Network
  - ❖ No graph diffusion
- ○ NDCN without control parameter W
  - ❖ Determined dynamics



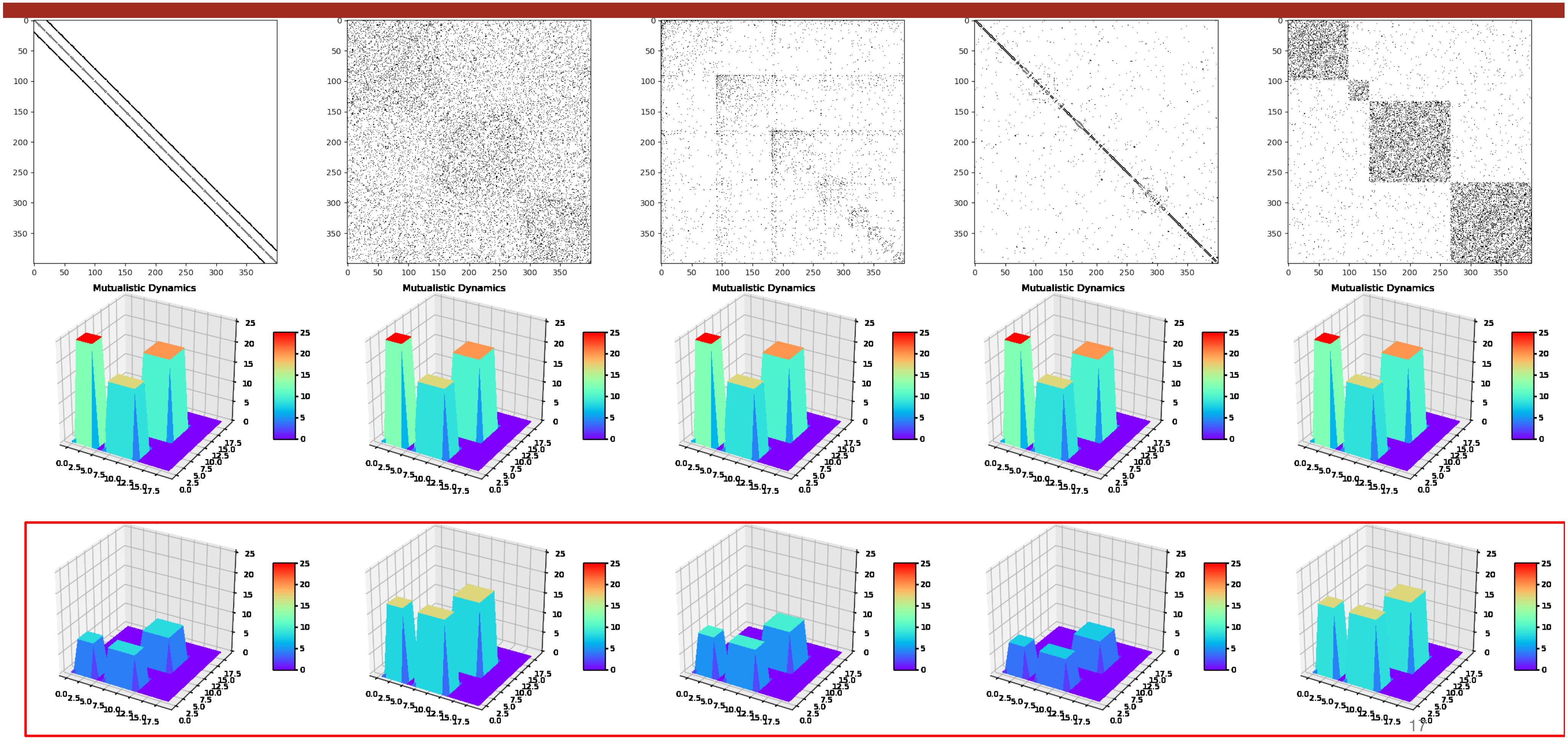
$$\begin{aligned}
\underset{W_*, b_*}{\operatorname{argmin}} \quad & \mathcal{L} = \int_0^T |X(t) - \hat{X(t)}|\, dt \\
\text{subject to} \quad & X_h(t) = \tanh\Big(X(t)W_e + b_e\Big)W_0 + b_0 \\
& \frac{dX_h(t)}{dt} = \operatorname{ReLU}\Big(\Phi X_h(t)W + b\Big), X_h(0) \\
& X(t) = X_h(t)W_d + b_d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

Chen et al. 2019. Neural Ordinary Differential Equations. *NeurIPS*.

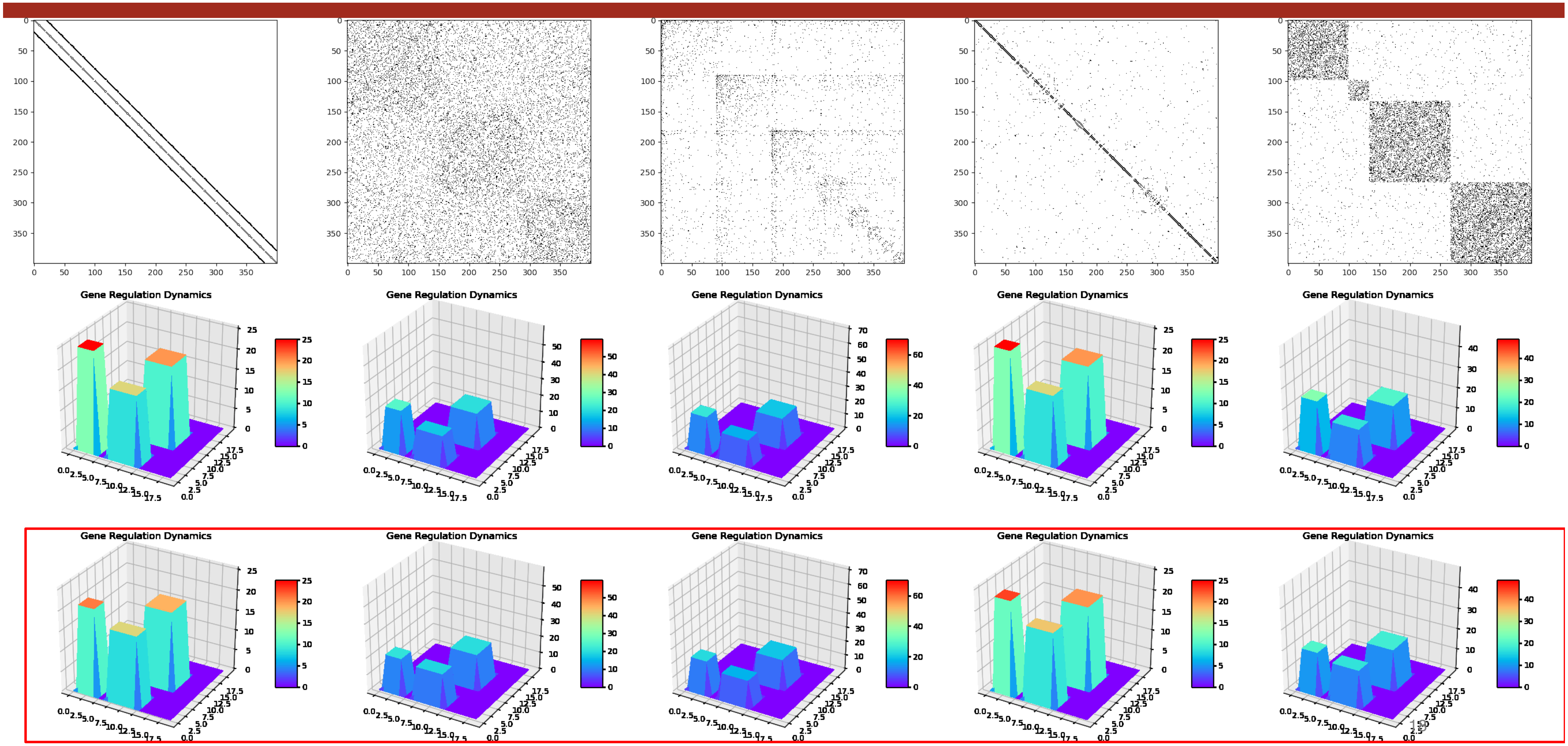
Exp1: Heat Diffusion on Different Graphs



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 ℓ_1 error with standard deviation (in percentage %) from 20 runs for 3 dynamics on 5 networks by each method.

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

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 ℓ_1 error with standard deviation (in percentage %) from 20 runs for 3 dynamics on 5 networks by each method.

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

Exp2: Structured Sequence Prediction

❑ The Problem (Structured sequence prediction):

- Input: $\left\{\widehat{X[1]}, \widehat{X[2]}, \ldots, \widehat{X[T]} \middle| 0 \leq 1 < \cdots < T\right\}, 1, .. T$ 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]=LSTM(GCN([t], G))
  - GRU-GNN
    - X[t+1]=GRU(GCN([t], G))
  - RNN-GNN
    - X[t+1]=RNN(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 ℓ_1 error with standard deviation (in percentage $\%$) from 20 runs for 3 dynamics on 5 networks by each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	LSTM-GNN	12.8 ± 2.1	21.6 ± 7.7	12.4 ± 5.1	11.6 ± 2.2	13.5 ± 4.2
	GRU-GNN	11.2 ± 2.2	9.1 ± 2.3	8.8 ± 1.3	9.3 ± 1.7	7.9 ± 0.8
	RNN-GNN	18.8 ± 5.9	25.0 ± 5.6	18.9 ± 6.5	21.8 ± 3.8	16.1 ± 0.0
	NDCN	$\mathbf{4.3 \pm 0.7}$	$\mathbf{4.7 \pm 1.7}$	$\mathbf{5.4 \pm 0.4}$	$\mathbf{2.7 \pm 0.4}$	$\mathbf{5.3 \pm 0.7}$
Mutualistic Interaction	LSTM-GNN	51.4 ± 3.3	24.2 ± 24.2	27.0 ± 7.1	58.2 ± 2.4	25.0 ± 22.3
	GRU-GNN	49.8 ± 4.1	$\mathbf{1.0 \pm 3.6}$	12.2 ± 0.8	51.1 ± 4.7	$\mathbf{3.7 \pm 4.0}$
	RNN-GNN	56.6 ± 0.1	8.4 ± 11.3	12.0 ± 0.4	57.4 ± 1.9	8.2 ± 6.4
	NDCN	$\mathbf{29.8 \pm 1.6}$	$\mathbf{4.7 \pm 1.1}$	$\mathbf{11.2 \pm 5.0}$	$\mathbf{15.9 \pm 2.2}$	$\mathbf{3.8 \pm 0.9}$
Gene Regulation	LSTM-GNN	27.7 ± 3.2	67.3 ± 14.2	38.8 ± 12.7	13.1 ± 2.0	53.1 ± 16.4
	GRU-GNN	24.2 ± 2.8	50.9 ± 6.4	35.1 ± 15.1	11.1 ± 1.8	46.2 ± 7.6
	RNN-GNN	28.0 ± 6.8	56.5 ± 5.7	42.0 ± 12.8	14.0 ± 5.3	46.5 ± 3.5
	NDCN	$\mathbf{18.6 \pm 9.9}$	$\mathbf{2.4 \pm 0.9}$	$\mathbf{4.1 \pm 1.4}$	$\mathbf{5.5 \pm 0.8}$	$\mathbf{2.9 \pm 0.5}$

Exp3. Node Semi-suprvised Classification

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.

Dataset	N	E	D	C	Train/Valid/Test
Cora	2,708	5,429	1,433	7	140/500/1,000
Citeseer	3,327	4,732	3,703	6	120/500/1,000
Pubmed	19,717	44,338	500	3	60/500/1,000

Exp3. Node Semi-suprvised Classification

❑ Baselines

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

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\, W^{(1)}\Big)$$

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

Kipf et al. 2016. Semi-Supervised Classification with Graph Convolutional Networks
Velickovic et al. 2017. Graph Attention Networks

Exp3. Node Semi-suprvised Classification

Interpretation of model

- Input: G, $[X, Mask \odot Y]$, features and some node labels
- Output: To Complete Y
- Model: A graph dynamics to spread features and labels over time T
  - $\frac{d[X,Y]}{dt} = f(G$

$$
\begin{aligned}
\underset{W_e, b_e, W_d, b_d}{\operatorname{argmin}} \quad & \mathcal{L} = \int_0^T \mathcal{R}(t)\, dt - \sum_{i=1}^{n} \sum_{k=1}^{c} \hat{Y}_{i,k}(T) \log Y_{i,k}(T) \\
\text{subject to} \quad & X_h(0) = \tanh\left(X(0)W_e + b_e\right) \\
& \frac{dX_h(t)}{dt} = \mathrm{ReLU}\left(\Phi X_h(t)\right) \\
& Y(T) = \mathrm{softmax}(X_h(T)W_d + b_d)
\end{aligned}
$$

Exp3. Node Semi-suprvised 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.

Model	Cora	Citeseer	Pubmed
GCN	81.5	70.3	79.0
AGNN	83.1 ± 0.1	71.7 ± 0.1	$\mathbf{79.9 \pm 0.1}$
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
NDCN	$\mathbf{83.3 \pm 0.6}$	$\mathbf{73.1 \pm 0.6}$	$\mathbf{79.8 \pm 0.4}$

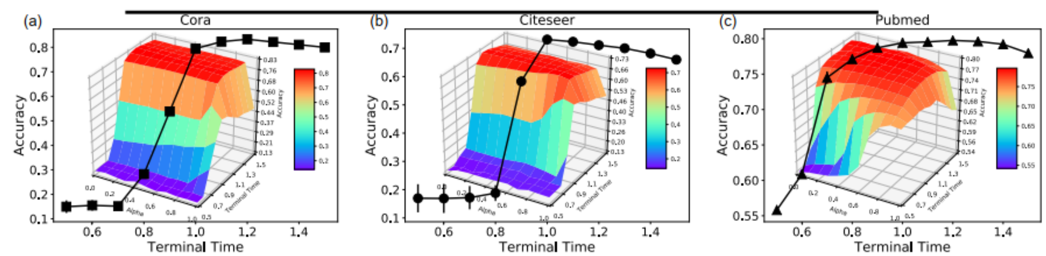


Figure 5: *Our NDCN model captures continuous-time dynamics.* Mean classification accuracy of 100 runs over terminal time when given a specific α. Insets are the accuracy over the two-dimensional space of terminal time and α

Summary

- **Our NDCN, a unified framework to solve**
  - Continuous-time network dynamics prediction:
  - Structured sequence prediction
  - Node regression/classification at final state

  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