CSDS 452 Causality and Machine Learning

Lecture 11: Causal Effect on Graphs

Instructor: Jing Ma

Fall 2024, CDS@CWRU

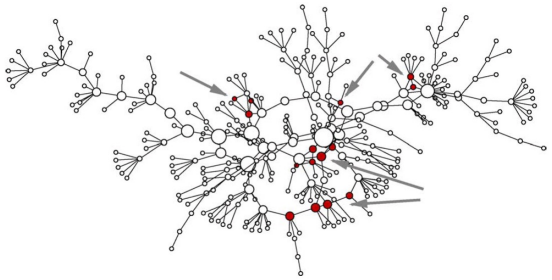
Outline

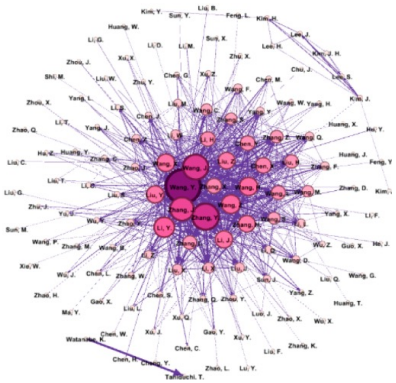
- Graph and Graph Neural Networks (GNNs)
- Deconfounding in Graphs
- Interference in Graphs

Graphs

Graphs have been extensively used for modeling many real-world systems with connected units

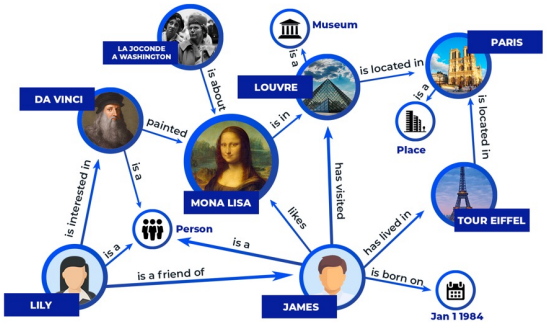
Social Network
[Newman 2005]





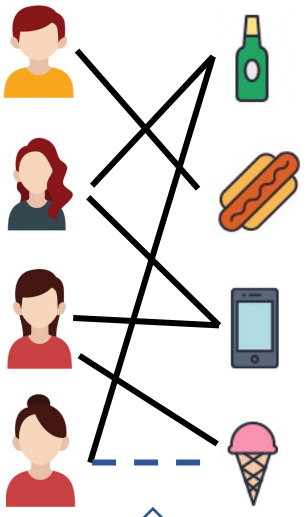
Cooperation Network
[Zhang, 2019]

Knowledge Graph
[Seth, 2019]

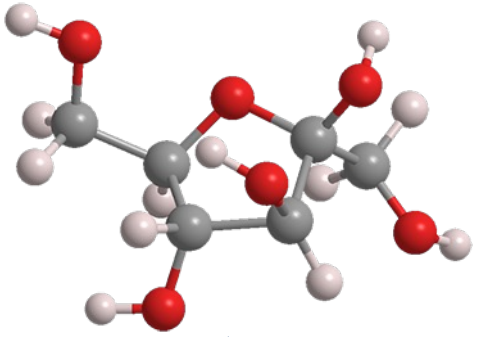


Machine Learning (ML) on Graphs

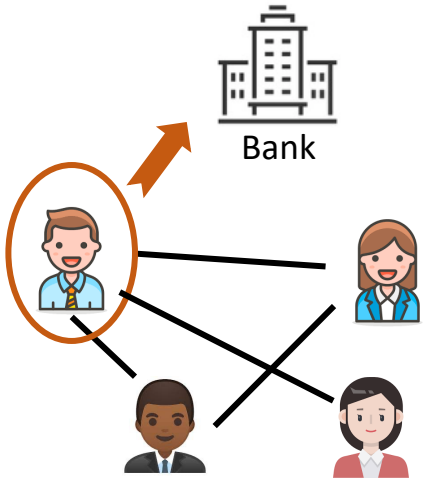
- ML has been widely used in different tasks (e.g., node classification) on graphs with many applications (e.g., recommender system, medicine & health, economics)
- However, in many cases, we are more interested in the causality inside data.



How much does this ad campaign influence the user's decision?



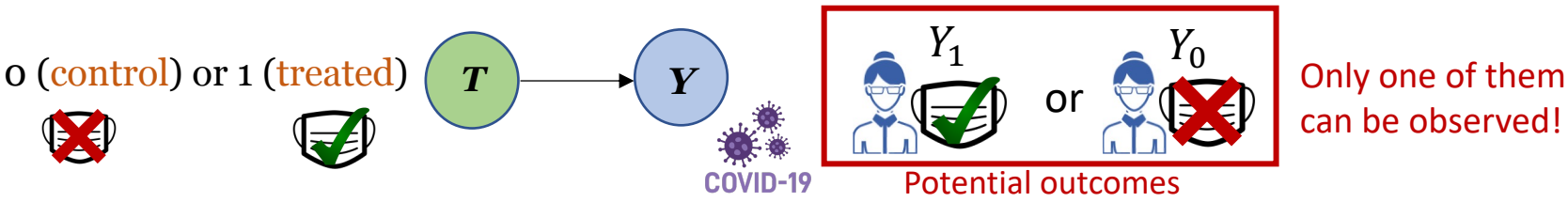
Which substructure causes a certain molecular property?



Will my credit application get approved if I intervene on my previous transaction ?

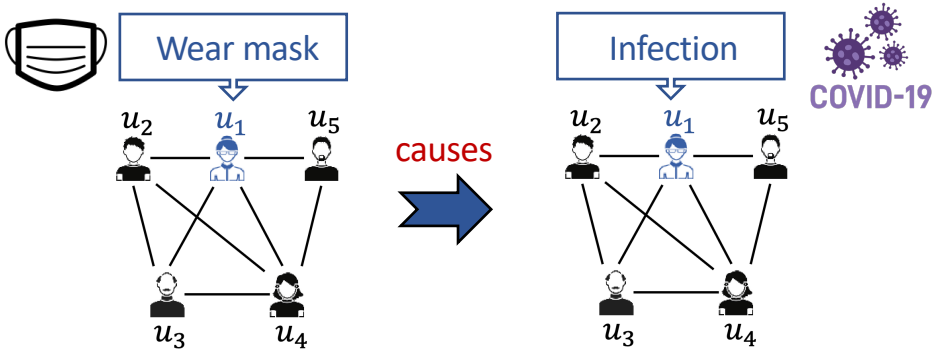
Causal Inference on Graphs

- Causal inference studies the causal relations rather than statistical dependencies between variables
- **Causal effect estimation**: assessing the causal effects of a treatment on an outcome for one/a group of units



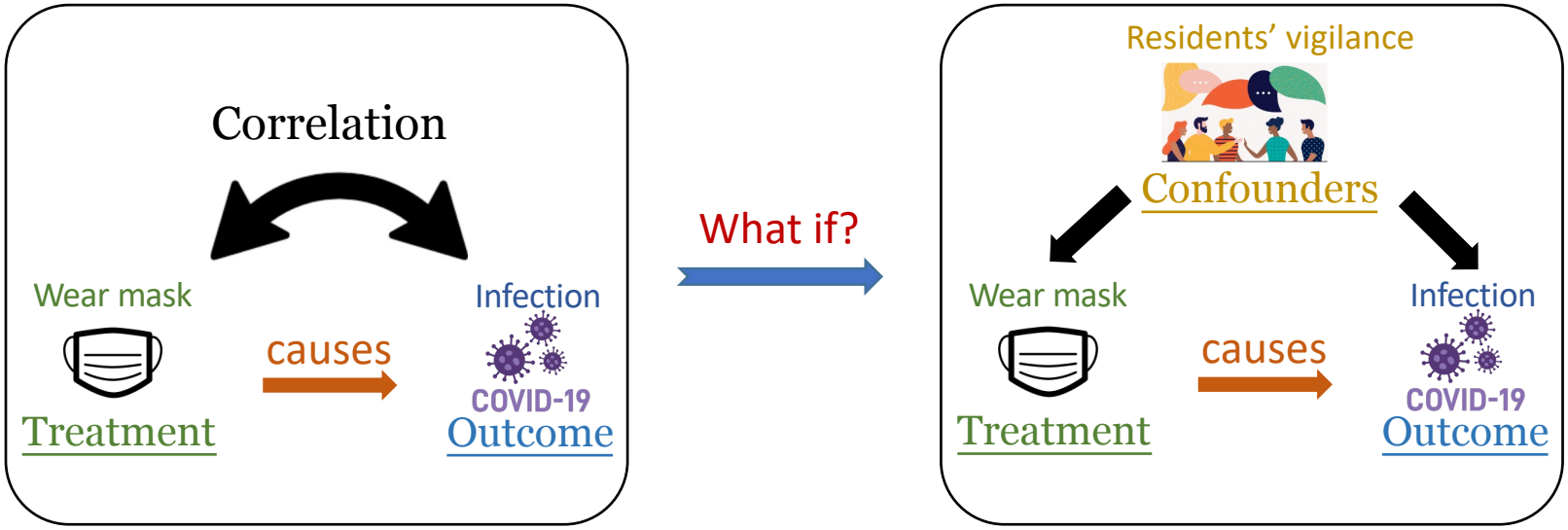
- **Causal effect estimation on graphs** Causal effect = Y_1 - Y_0

  Question: In a contact network, how does the usage of face mask influence COVID-19 infection risk?



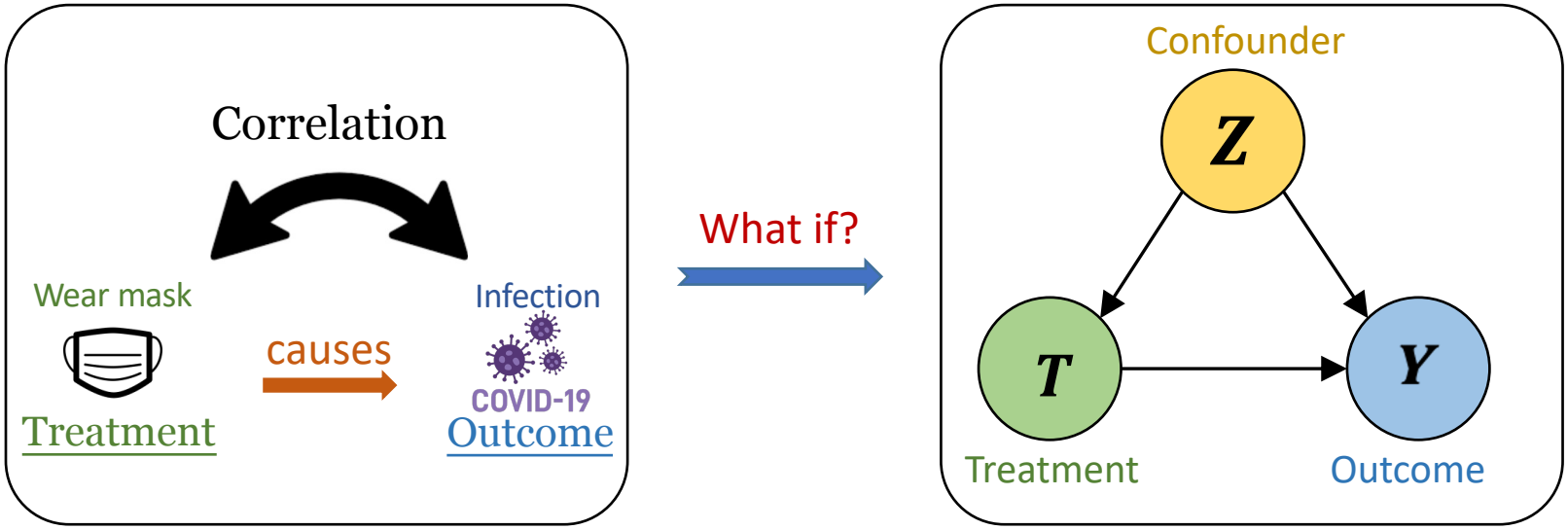
Challenge 1: Hidden Confounders

- Correlation ≠ Causation



Challenge 1: Hidden Confounders

- Correlation ≠ Causation

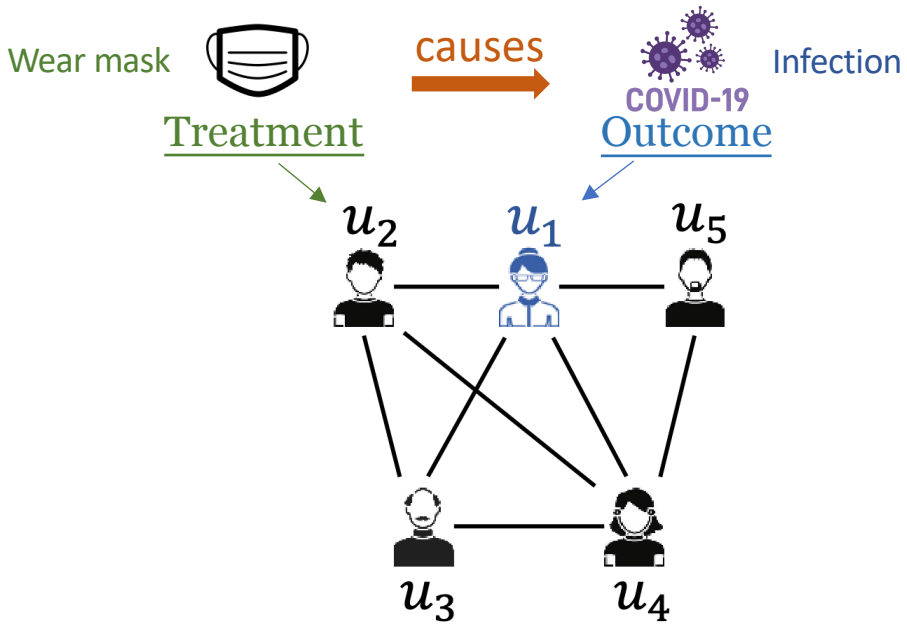


- Confounders influence both the treatment and the outcome
- If not properly handled, confounders can bring biases to causal effect estimation
- The confounders are often unobserved
- The confounders may be time-varying

Hard to control for these confounders

Challenge 2: Graph Interference

- **Interference**: the treatment of an individual may causally affect the outcome of other individuals
  - **Example**: whether a person wears a face mask in public may influence the infection risk of other people
- Traditional causal inference assumes interference does not exist, but interference is ubiquitous in networked data



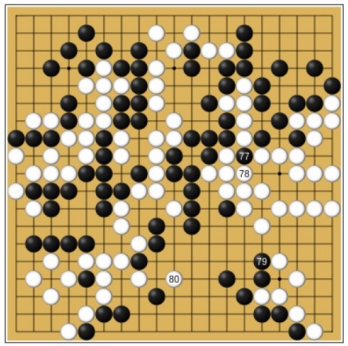
Outline

- Graph and Graph Neural Networks (GNNs)
- Deconfounding in Graphs
- Interference in Graphs

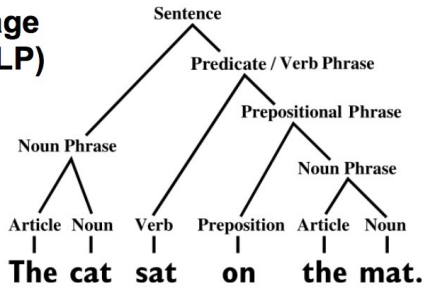
Traditional Neural Networks

Speech data

Grid games



Natural language processing (NLP)



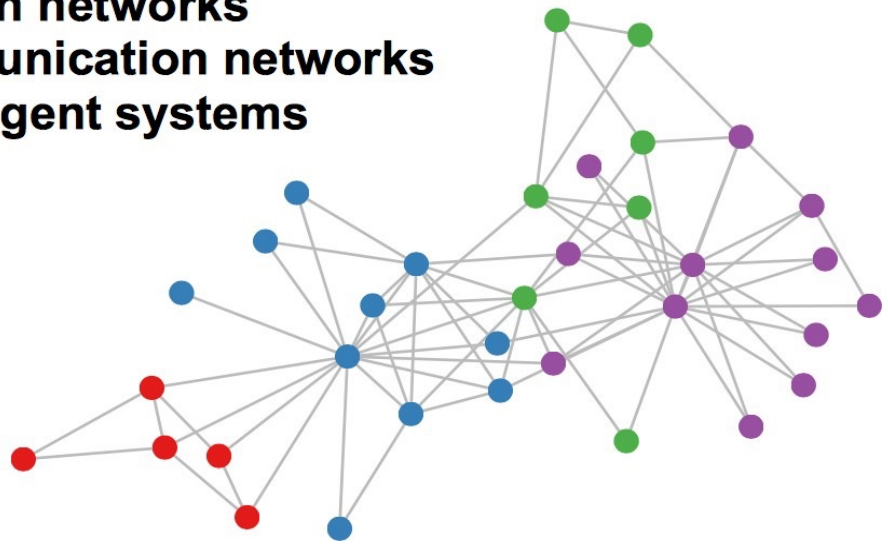
...

Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality

Graph-structured Data

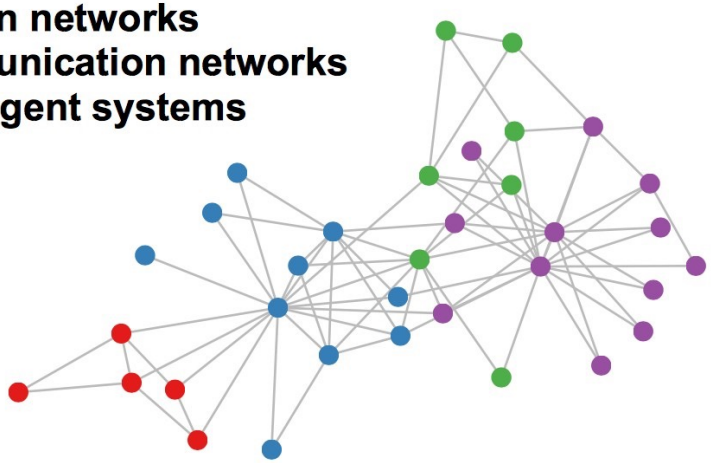
Social networks
Citation networks
Communication networks
Multi-agent systems



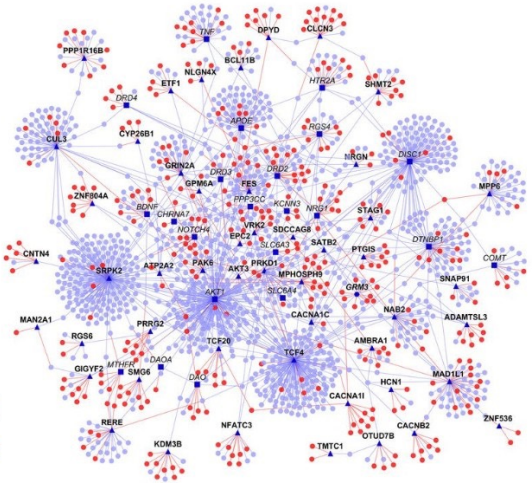
Slides from Thomas Kipf, **University of Amsterdam**

Graph-structured Data

Social networks
Citation networks
Communication networks
Multi-agent systems



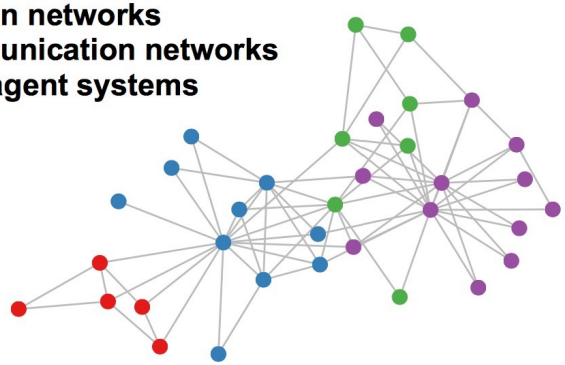
Protein interaction networks



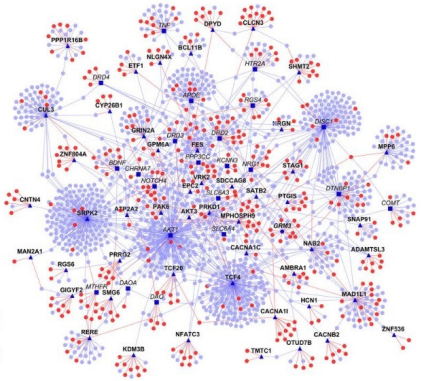
*slide from Thomas Kipf, **University of Amsterdam**

Graph-structured Data

Social networks
Citation networks
Communication networks
Multi-agent systems



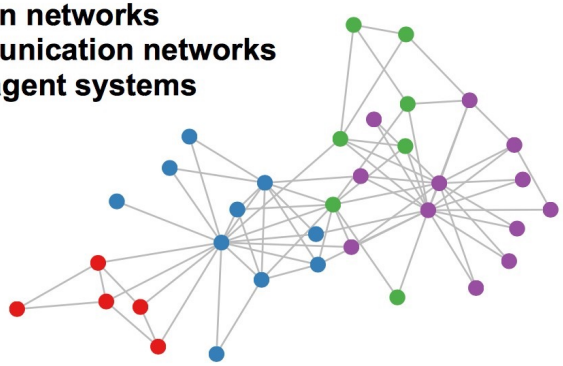
Protein interaction networks

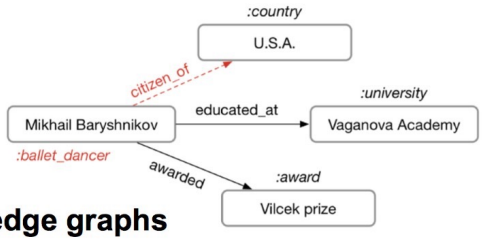


*slide from Thomas Kipf, **University of Amsterdam**

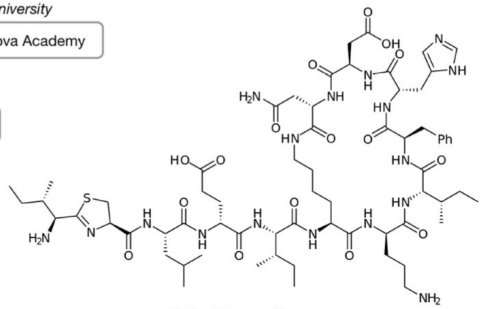
Graph-structured Data

Social networks
Citation networks
Communication networks
Multi-agent systems

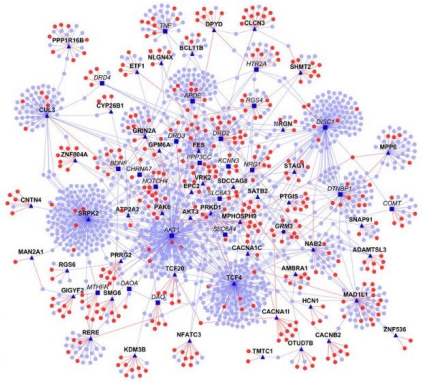




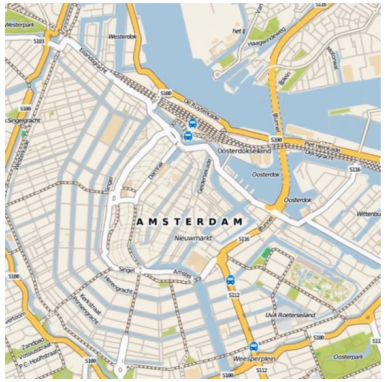
Knowledge graphs



Molecules



Protein interaction networks

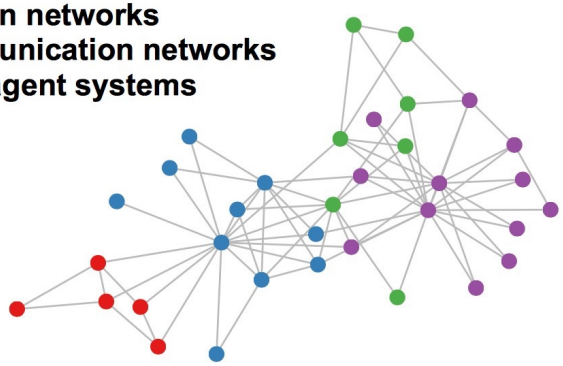


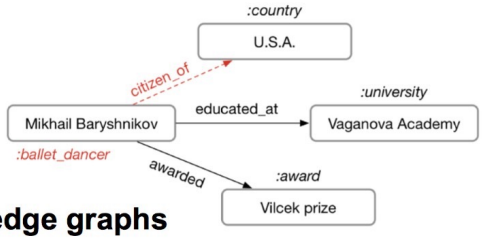
Road maps

*slide from Thomas Kipf, **University of Amsterdam**

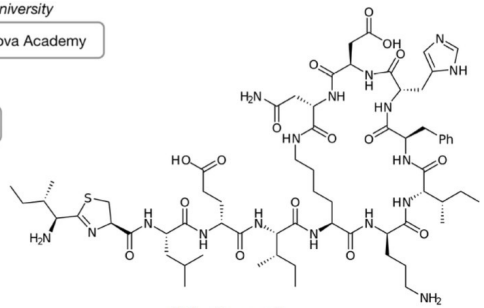
Graph-structured Data

Social networks
Citation networks
Communication networks
Multi-agent systems

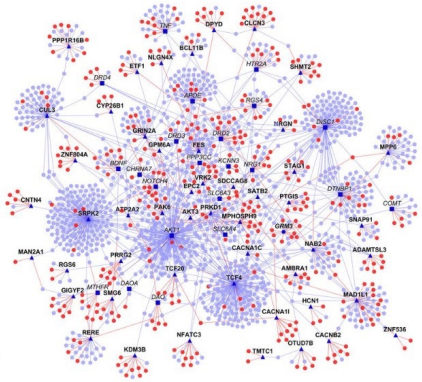




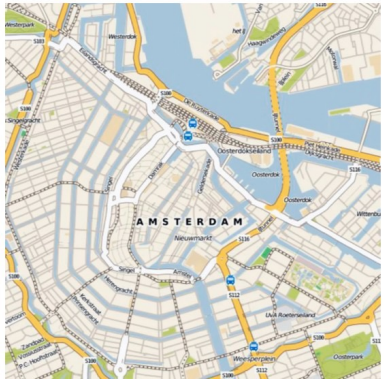
Knowledge graphs



Molecules



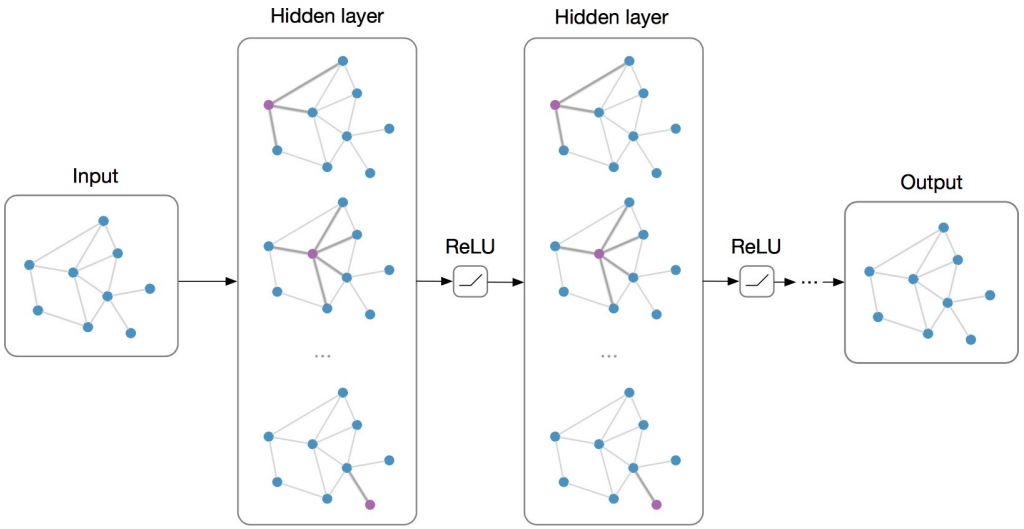
Protein interaction networks



Road maps

Standard **CNN** and **RNN** architectures don't work on this data

Graph Neural Networks (GNNs)



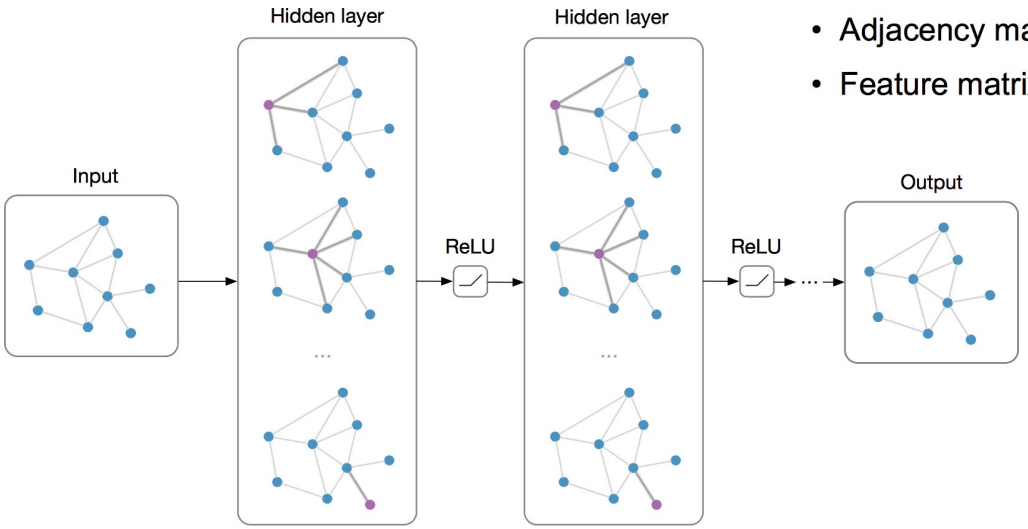
Main Idea: Pass messages across graph structure

Alternative Interpretation: Pass messages between nodes to refine node (and possibly edge) representations

Graph Neural Networks (GNNs)

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

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

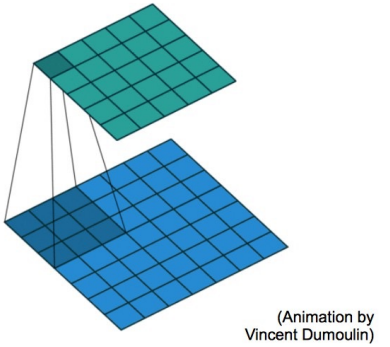


Main Idea: Pass messages across graph structure

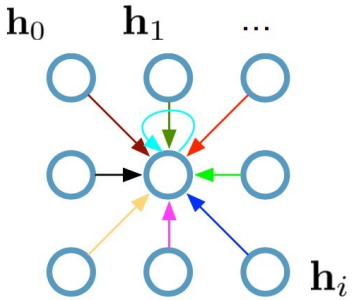
Alternative Interpretation: Pass massages between nodes to refine node (and possibly edge) representations

Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:

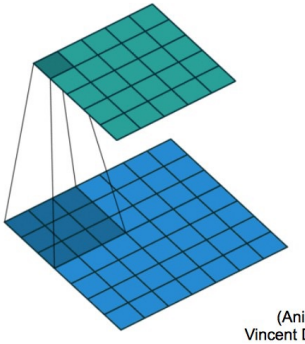


(Animation by Vincent Dumoulin)

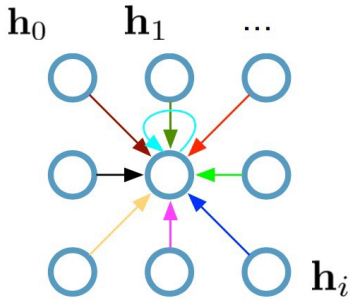


Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:



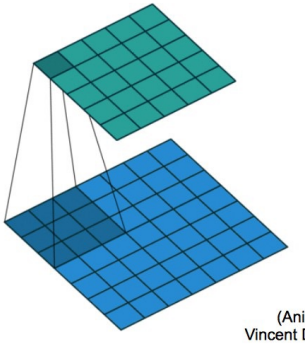
(Animation by Vincent Dumoulin)



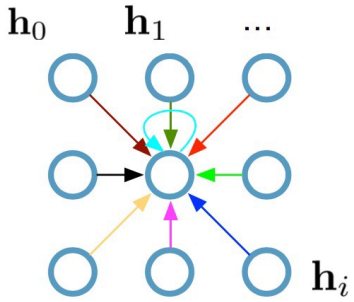
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



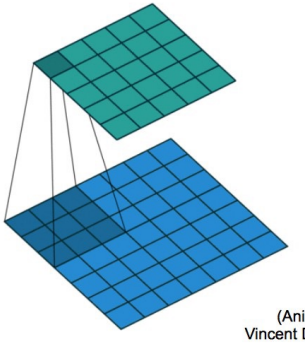
Update for a single pixel:

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

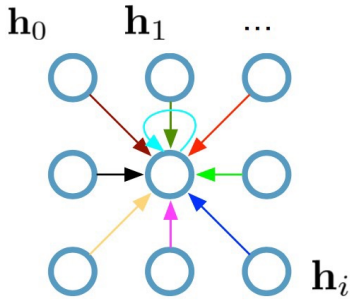
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

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

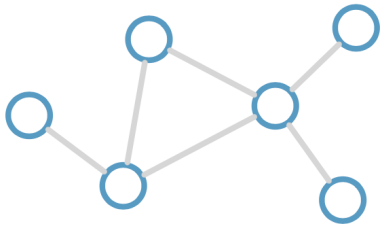
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

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

Graph Convolutional Networks (GCNs)

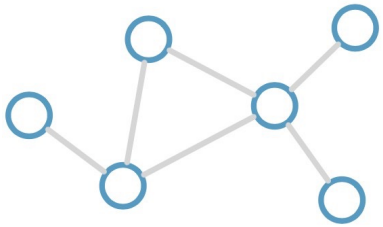
Consider this undirected graph:



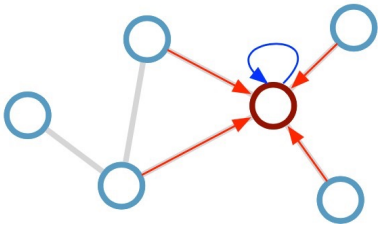
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Graph Convolutional Networks (GCNs)

Consider this undirected graph:

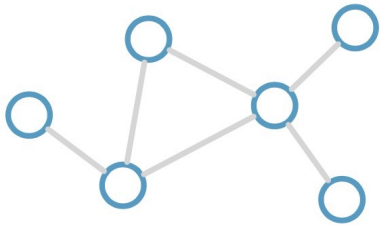


Calculate update for node in red:

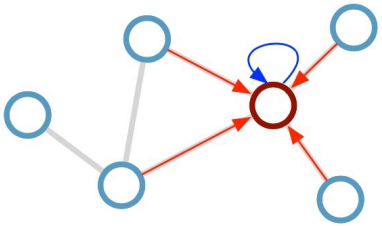


Graph Convolutional Networks (GCNs)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

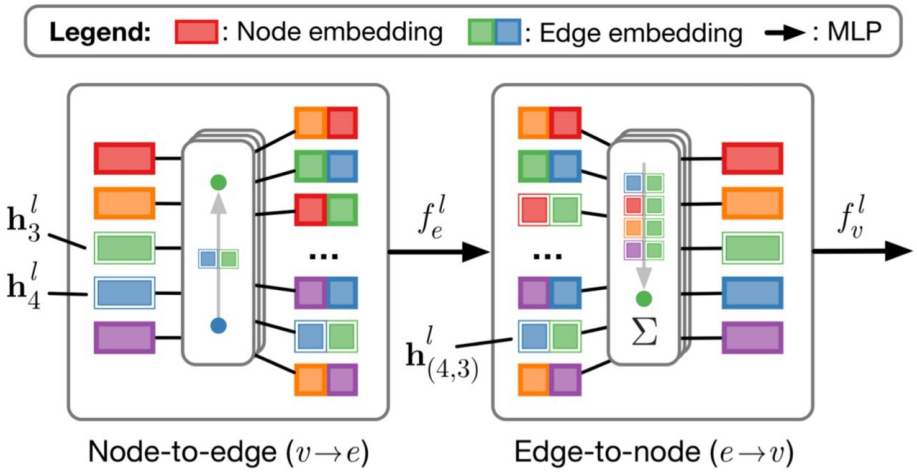
$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

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

$\mathcal{N}_i$: neighbor indices

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

GNNs with Edge Embeddings



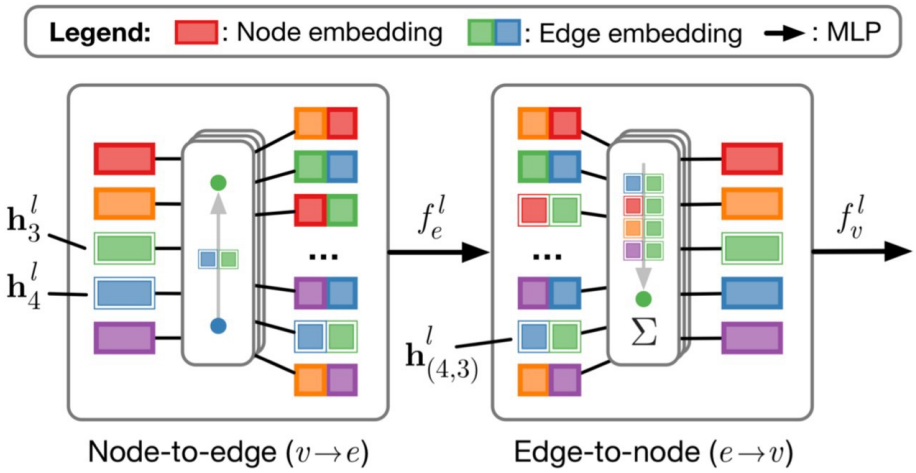
Formally:

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

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

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

GNNs with Edge Embeddings



Pros:

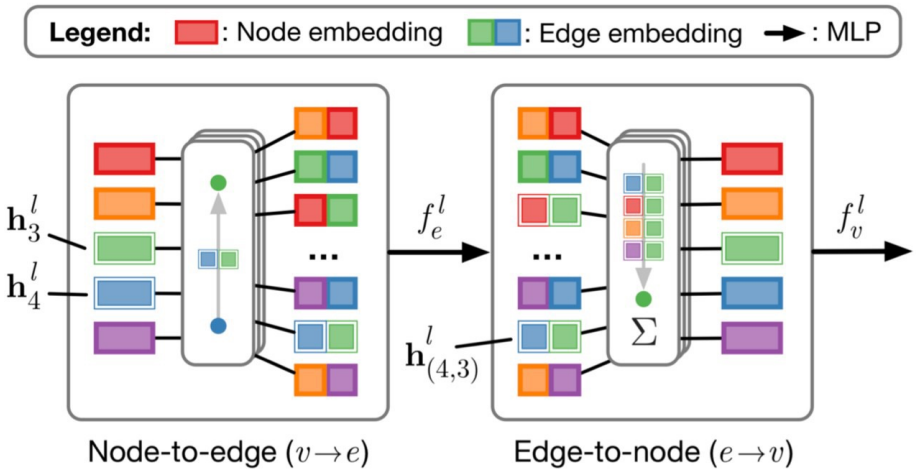
- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

Formally:

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

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

GNNs with Edge Embeddings



Formally:

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

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

Pros:

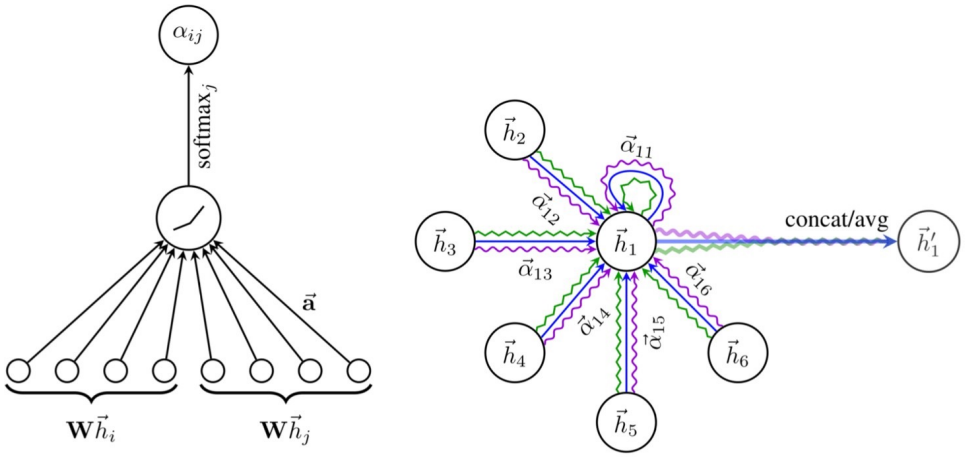
- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

Cons:

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

➡ In practice limited to small graphs

Graph Neural Networks (GNNs) with Attention

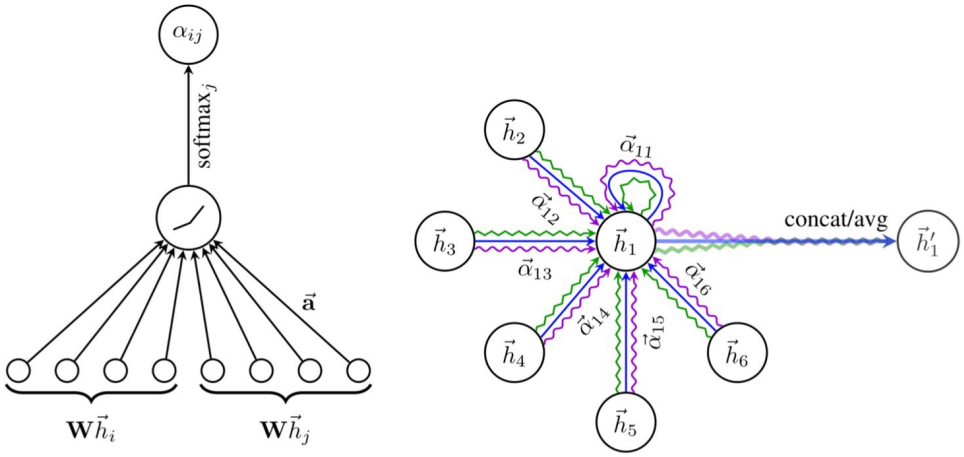


[Figure from Veličković et al. (ICLR 2018)]

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

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

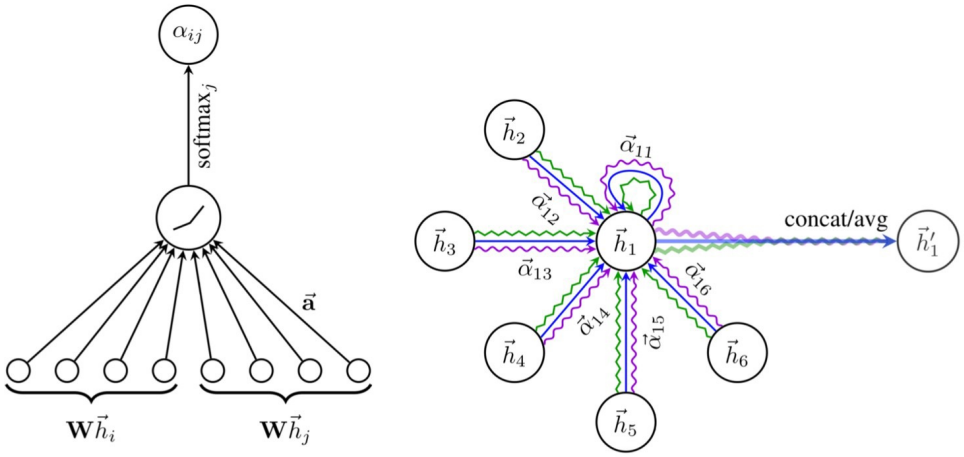
Graph Neural Networks (GNNs) with Attention



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

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

Graph Neural Networks (GNNs) with Attention



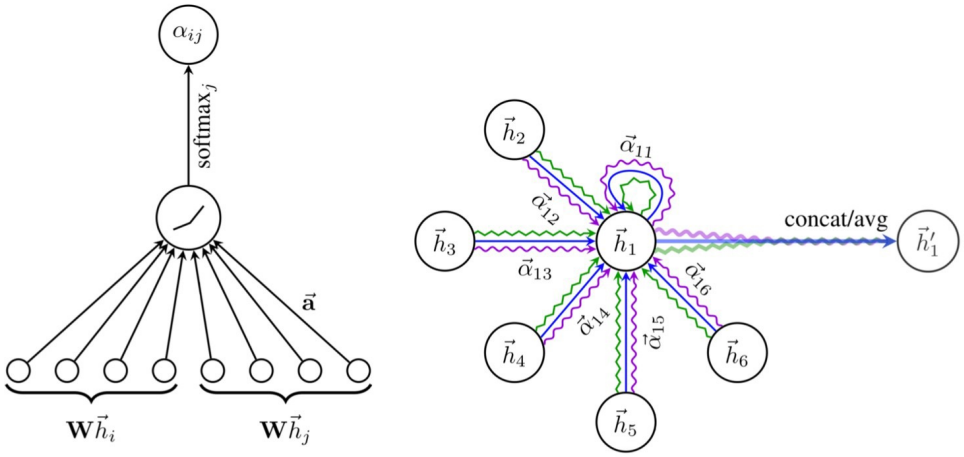
[Figure from Veličković et al. (ICLR 2018)]

Pros:

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

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

Graph Neural Networks (GNNs) with Attention



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

Pros:

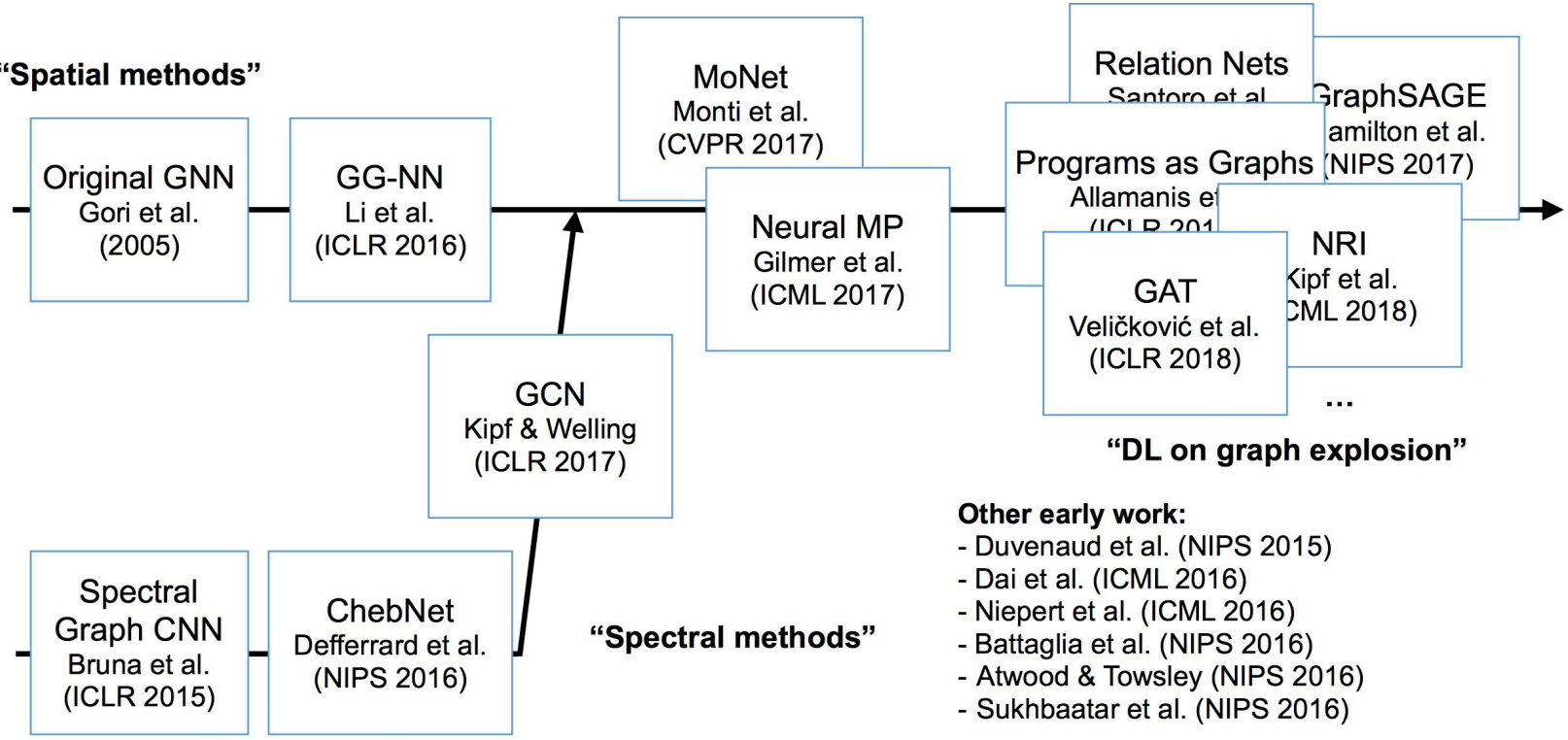
- No need to store intermediate edge-based activation vectors (when using dot-product attn.)
- Slower than GCNs but faster than GNNs with edge embeddings

Cons:

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

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

A Brief History of Graph Neural Nets



Other early work:

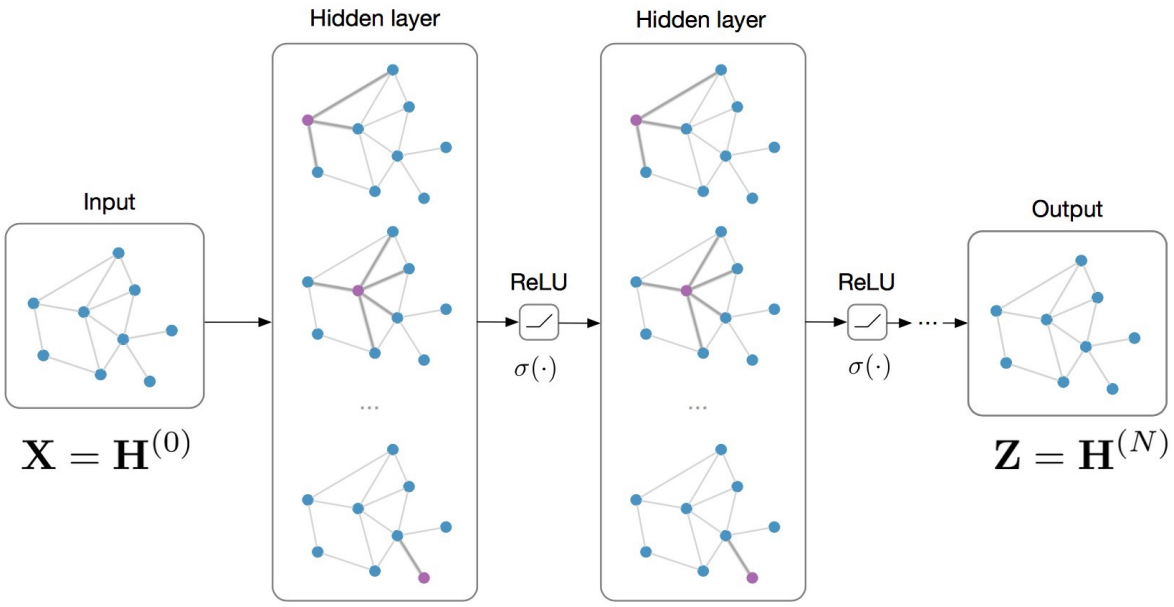
- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

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

How do we use GNN / GCN for real problems?

Classification and Link Prediction with GNNs / GCNs

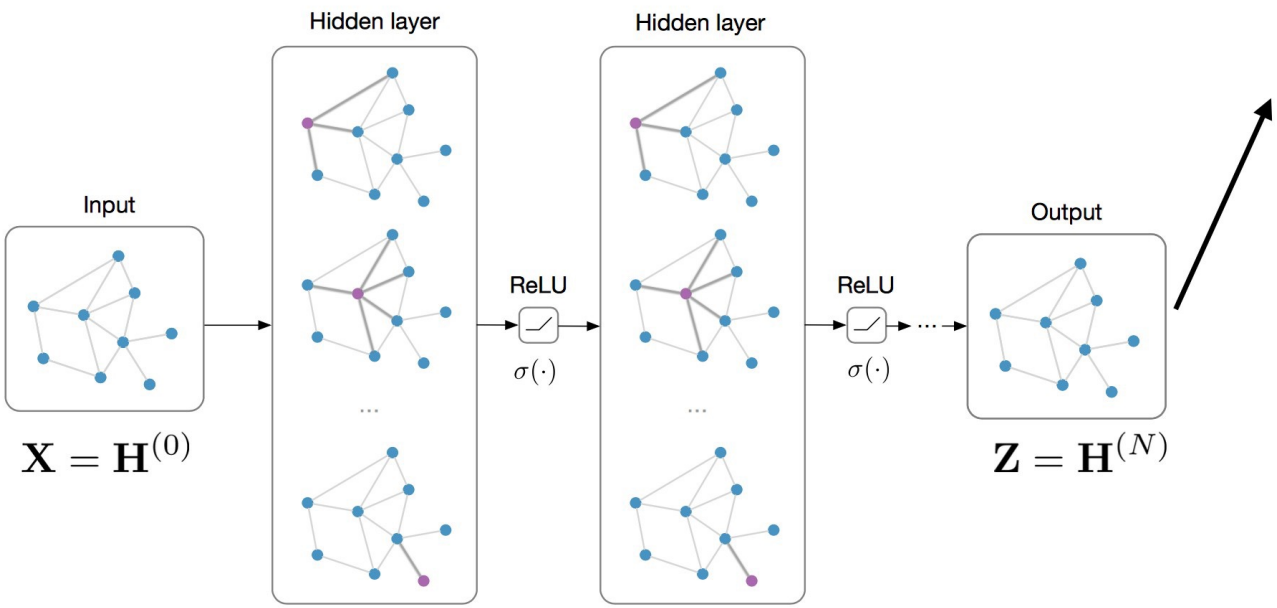
Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



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

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:

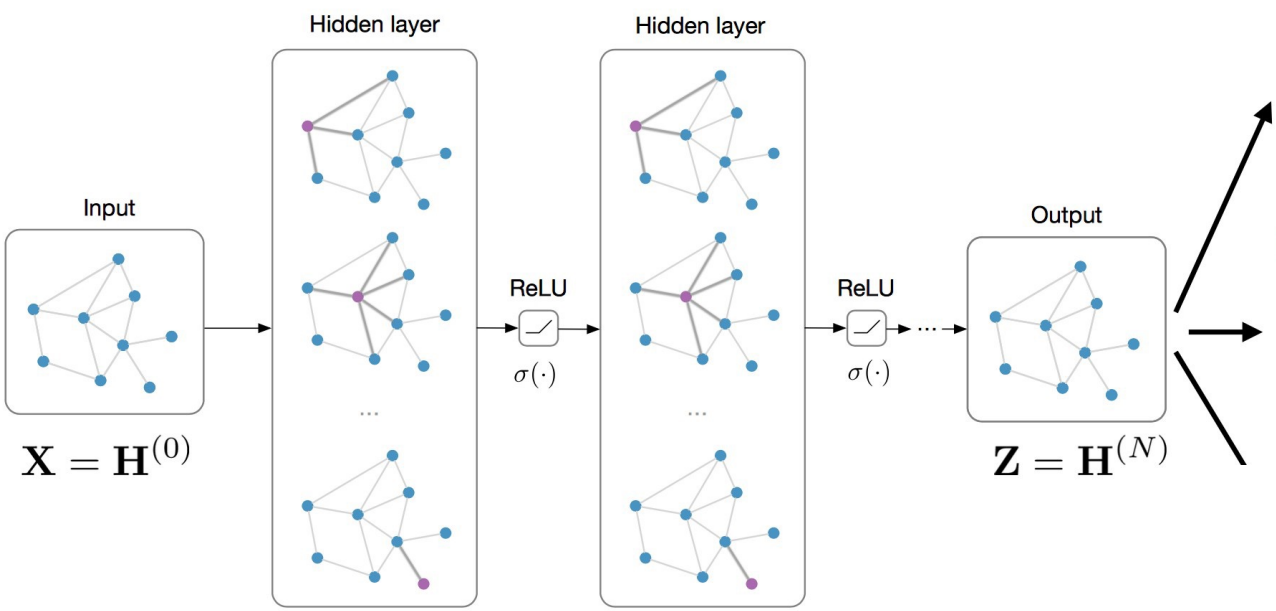
$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

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

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:

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

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

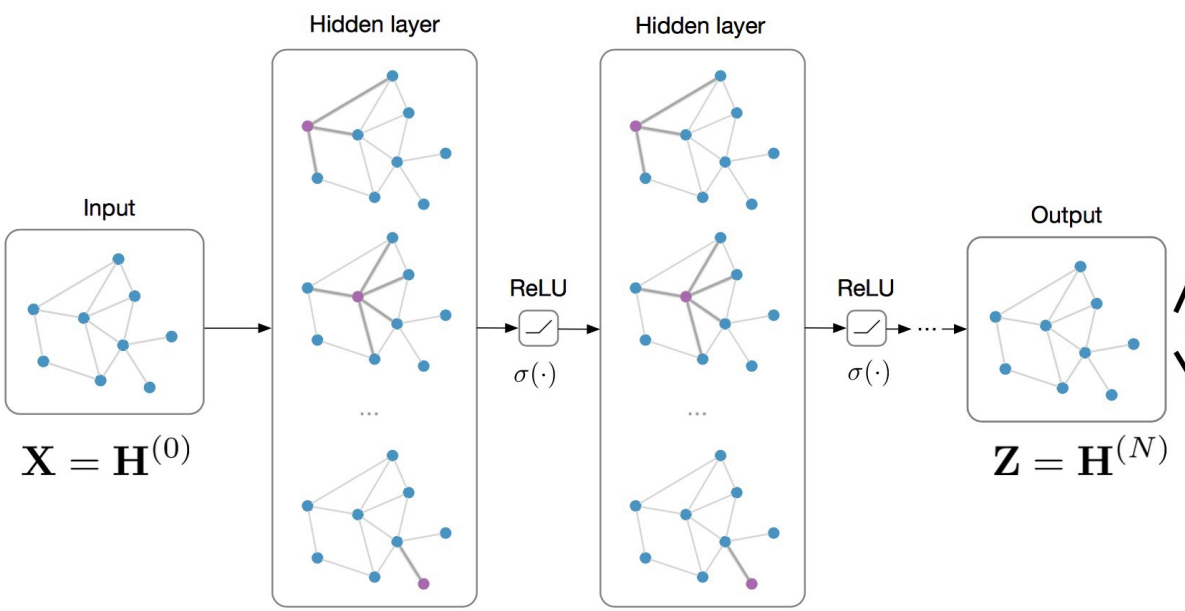
$\mathrm{softmax}(\sum_n \mathbf{z_n})$

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

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

Classification and Link Prediction with GNNs / GCNs

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



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

Node classification:

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

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

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

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"

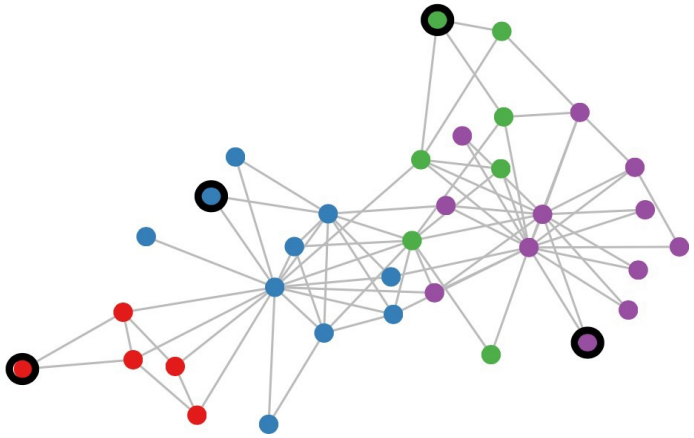
Semi-supervised Classification on Graphs

Setting:

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

Task:

Predict node label of unlabeled nodes



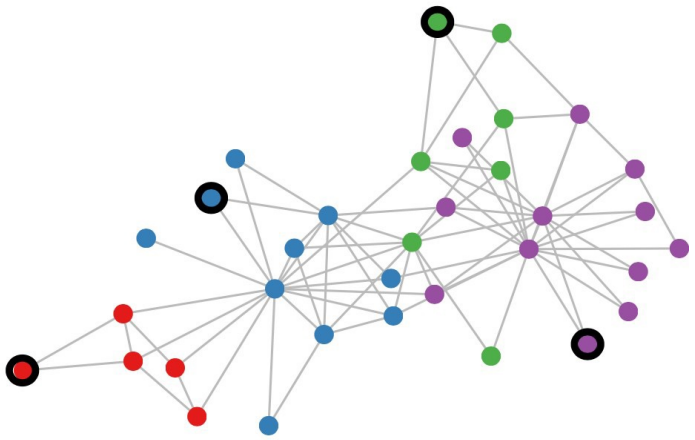
Semi-supervised Classification on Graphs

Setting:

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

Task:

Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

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

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

$\mathbf{Y}$ label matrix

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

Conclusions

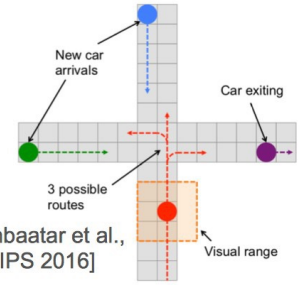
— **Deep learning on graphs works and is very effective!**

— Exciting area: lots of new applications and extensions (hard to keep 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]

Open problems:

— Theory

— Scalable, stable generative models

— Learning on large, evolving data

— Multi-modal and cross-model learning (e.g., sequence2graph)

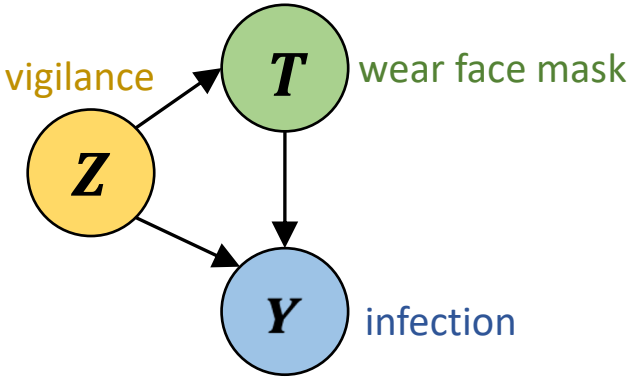
*slide from Thomas Kipf, **University of Amsterdam**

Outline

- Graph and Graph Neural Networks (GNNs)
- Deconfounding in Graphs
  - Static graph
  - Dynamic graph
- Interference in Graphs

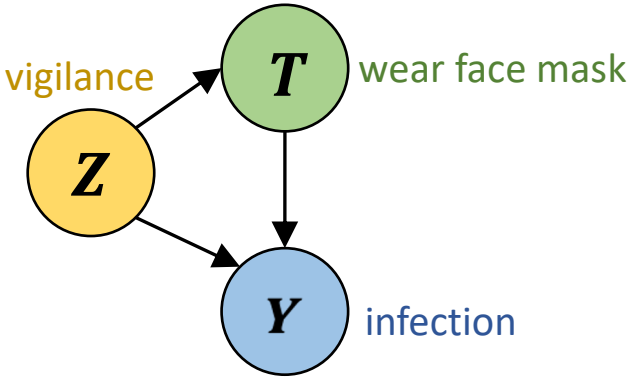
Causal Inference under Hidden Confounders

- Hidden confounders Z causally affect treatment T and outcome Y

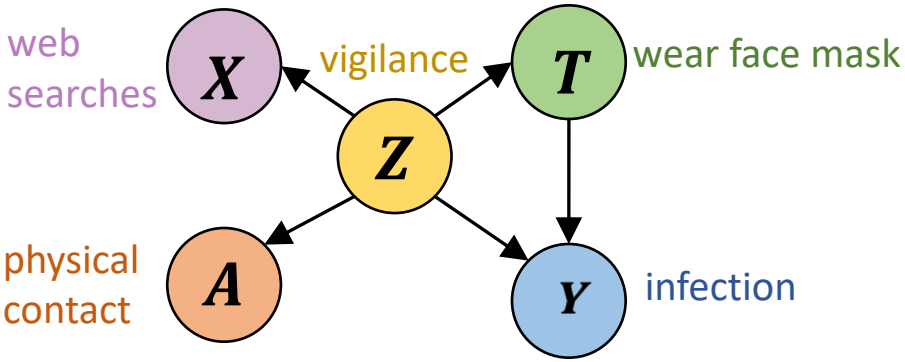


Causal Inference under Hidden Confounders

- Hidden confounders Z causally affect treatment T and outcome Y

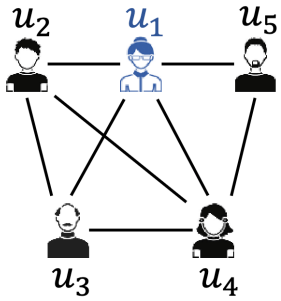


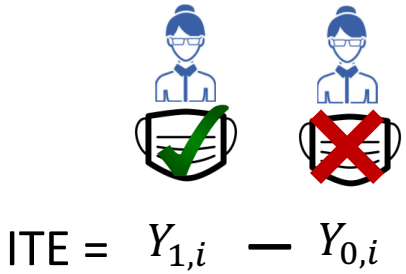
- Graph data (node features X and network structure A) can be used as <u>proxy variables</u> for hidden confounders Z



Key Idea

- **Motivation**: Hidden confounders often lead to biased causal effect estimation
- **Key idea**: Capture hidden confounders through representation learning from graph data





- Graph data can be proxy variables for hidden confounders
- Effective deep learning techniques are utilized
- Estimate ITE based on confounder representations

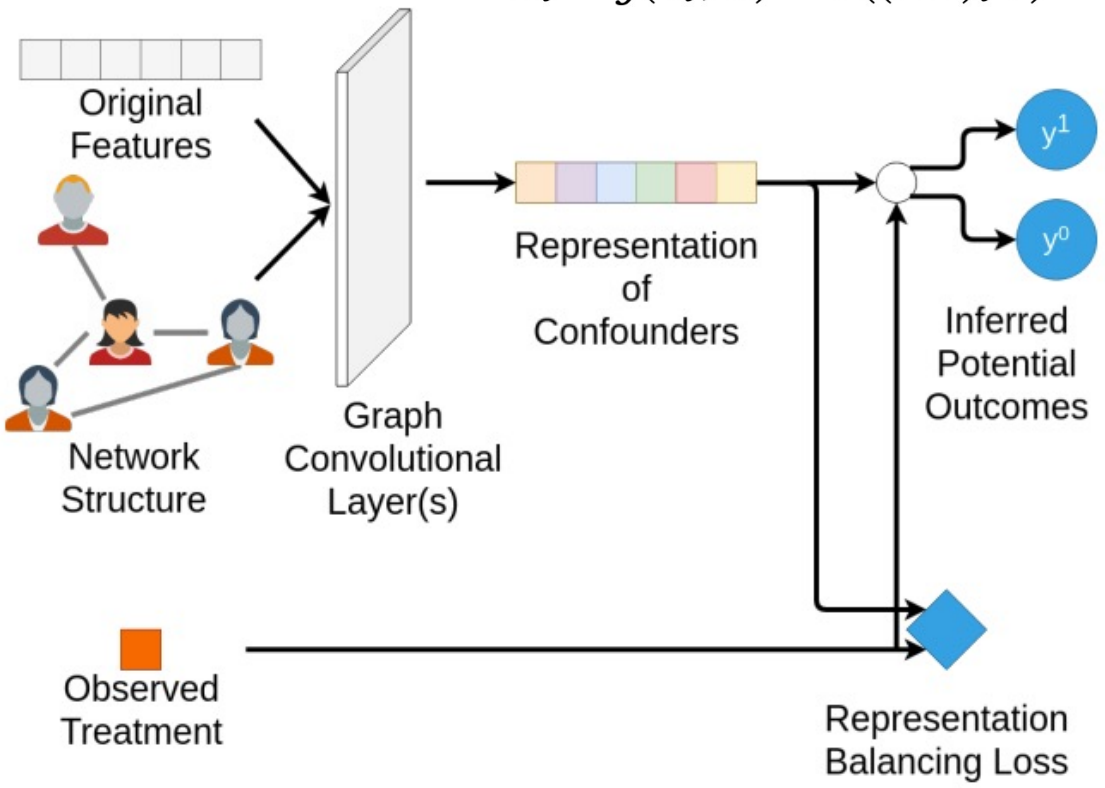
ITE defined on graph

Given the time-evolving networked observational data $\{\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}\}$, the goal is to learn the ITE τ_i for each instance i.

$$\tau_i = \tau(\mathbf{x}_i, \mathbf{A}) = \mathbb{E}[y_i^1 | \mathbf{x}_i, \mathbf{A}] - \mathbb{E}[y_i^0 | \mathbf{x}_i, \mathbf{A}]$$

Network deconfounder

$$\mathbf{h}_i = g(\mathbf{x}_i, \mathbf{A}) = \sigma((\hat{\mathbf{A}}\mathbf{X})_i\mathbf{U})$$



$$f(\mathbf{h}_i, t) = \begin{cases} f_1(\mathbf{h}_i) \text{ if } t = 1 \\ f_0(\mathbf{h}_i) \text{ if } t = 0 \end{cases}$$

Ruocheng Guo, Jundong Li, and Huan Liu. 2020. Learning Individual Causal Effects from Networked Observational Data. WSDM 2020.

Datasets & Simulation

Real-world data

	Instances	Edges	Features
BC	5,196	173,468	8,189
Flickr	7,575	239,738	12,047

Simulated data

κ_2	ATE Mean	STD
0.5	4.366	0.553
1	7.446	0.759
2	13.534	2.309
0.5	6.672	3.068
1	8.487	3.372
2	20.546	5.718

Simulate treatment

$$Pr(t = 1|\mathbf{x}_i, \mathbf{A}) = \frac{\exp(p_1^i)}{\exp(p_1^i) + \exp(p_0^i)};$$

$$p_1^i = \kappa_1 r(\mathbf{x}_i)^T r_1^c + \kappa_2 \sum_{j \in \mathcal{N}(i)} r(\mathbf{x}_j)^T r_1^c$$

$$= \kappa_1 r(\mathbf{x}_i)^T r_1^c + \kappa_2 (\mathbf{A} r(\mathbf{x}_j))^T r_1^c;$$

$$p_0^i = \kappa_1 r(\mathbf{x}_i)^T r_0^c + \kappa_2 \sum_{j \in \mathcal{N}(i)} r(\mathbf{x}_j)^T r_0^c$$

$$= \kappa_1 r(\mathbf{x}_i)^T r_0^c + \kappa_2 (\mathbf{A} r(\mathbf{x}_j))^T r_0^c,$$

Simulate potential outcomes

$$y^F(\mathbf{x}_i) = y_i = C(p_0^i + t_i p_1^i) + \epsilon;$$

$$y^{CF}(\mathbf{x}_i) = C[p_0^i + (1 - t_i) p_1^i] + \epsilon,$$

Experiment

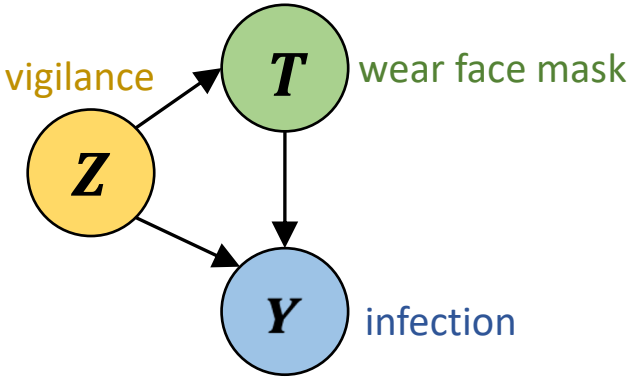
BlogCatalog						
κ_2	0.5		1		2	
	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}
NetDeconf (ours)	**4.532**	**0.979**	**4.597**	**0.984**	**9.532**	**2.130**
CFR-Wass	10.904	4.257	11.644	5.107	34.848	13.053
CFR-MMD	11.536	4.127	12.332	5.345	34.654	13.785
TARNet	11.570	4.228	13.561	8.170	34.420	13.122
CEVAE	7.481	1.279	10.387	1.998	24.215	5.566
Causal Forest	7.456	1.261	7.805	1.763	19.271	4.050
BART	4.808	2.680	5.770	2.278	11.608	6.418
Flickr						
	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}
NetDeconf (ours)	**4.286**	**0.805**	**5.789**	**1.359**	**9.817**	**2.700**
CFR-Wass	13.846	3.507	27.514	5.192	53.454	13.269
CFR-MMD	13.539	3.350	27.679	5.416	53.863	12.115
TARNet	14.329	3.389	28.466	5.978	55.066	13.105
CEVAE	12.099	1.732	22.496	4.415	42.985	5.393
Causal Forest	8.104	1.359	14.636	3.545	26.702	4.324
BART	4.907	2.323	9.517	6.548	13.155	9.643

Outline

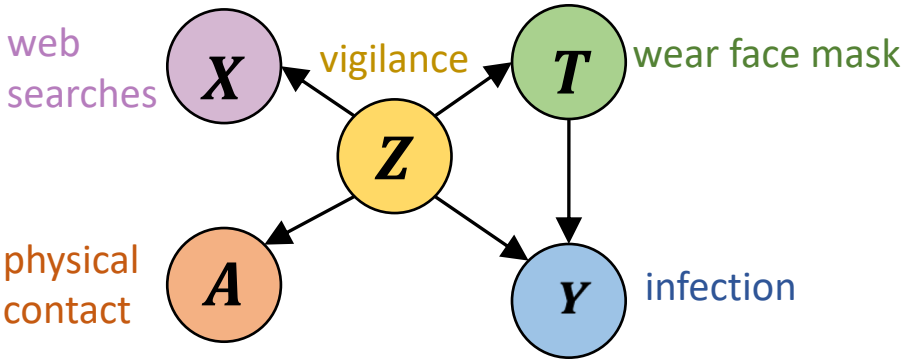
- Graph and Graph Neural Networks (GNNs)
- Deconfounding in Graphs
  - Static graph
  - Dynamic graph
- Interference in Graphs

Causal Inference under Hidden Confounders

- Hidden confounders Z causally affect treatment T and outcome Y

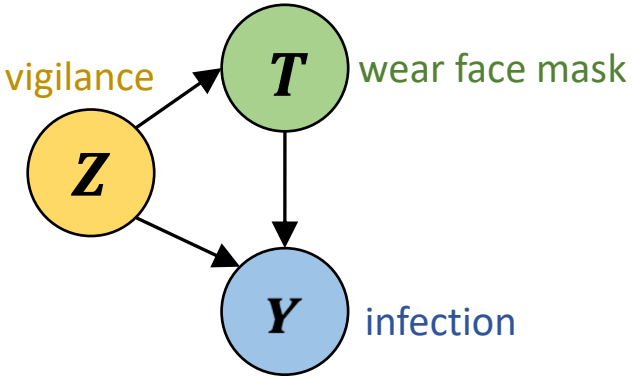


- Graph data (node features X and network structure A) can be used as <u>proxy variables</u> for hidden confounders Z

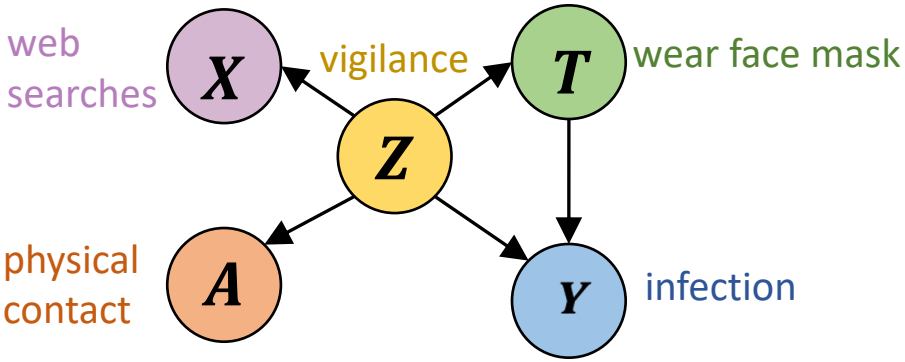


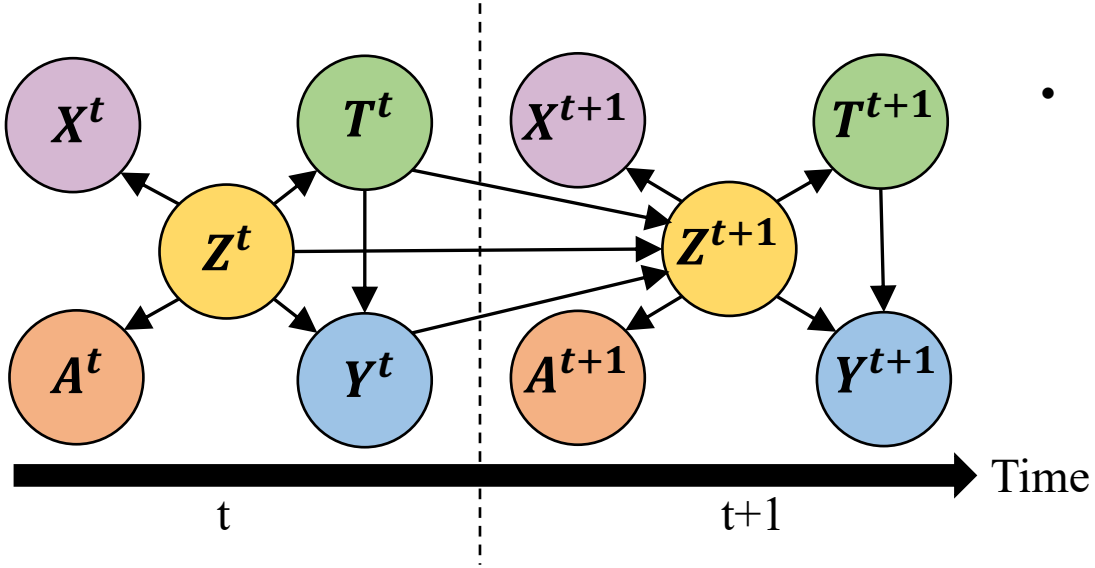
Causal Inference under Hidden Confounders

- Hidden confounders Z causally affect treatment T and outcome Y



- Graph data (node features X and network structure A) can be used as proxy variables for hidden confounders Z



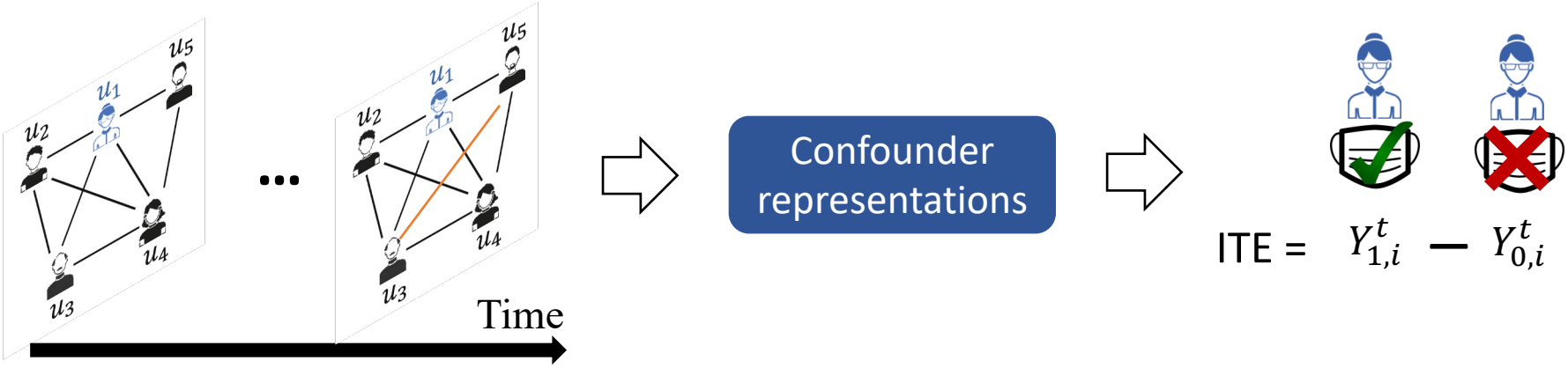


- Historical data (previous confounders Z^t, treatment T^t, outcome Y^t) can influence current confounders Z^{t+1};

Dynamic environment

Key Idea

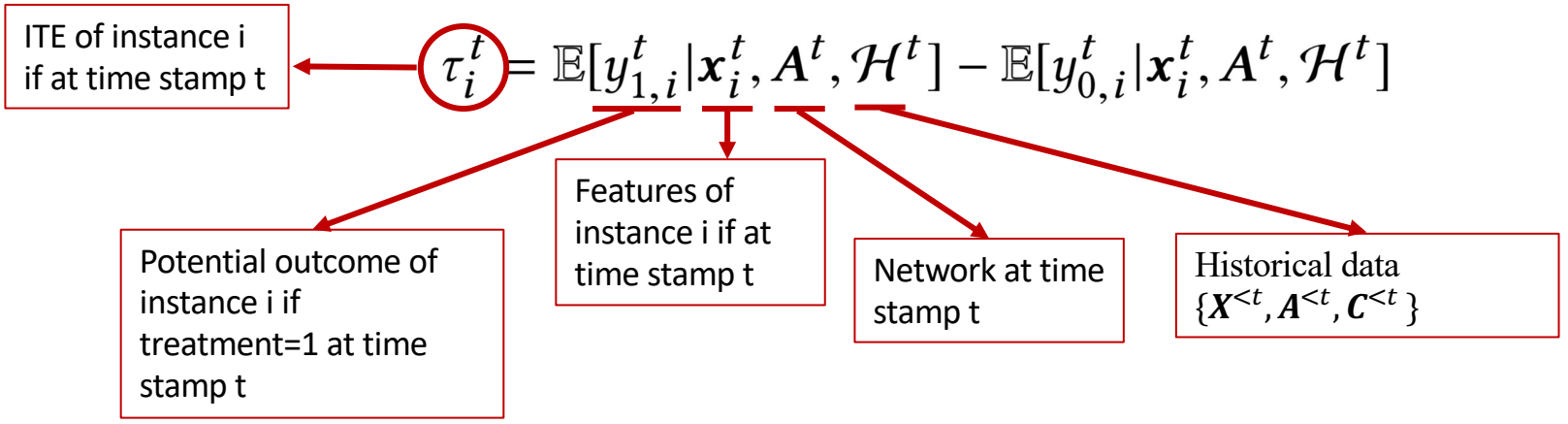
- **Motivation**: Hidden confounders often lead to biased causal effect estimation
- **Key idea**: Capture hidden confounders through representation learning from dynamic graph data



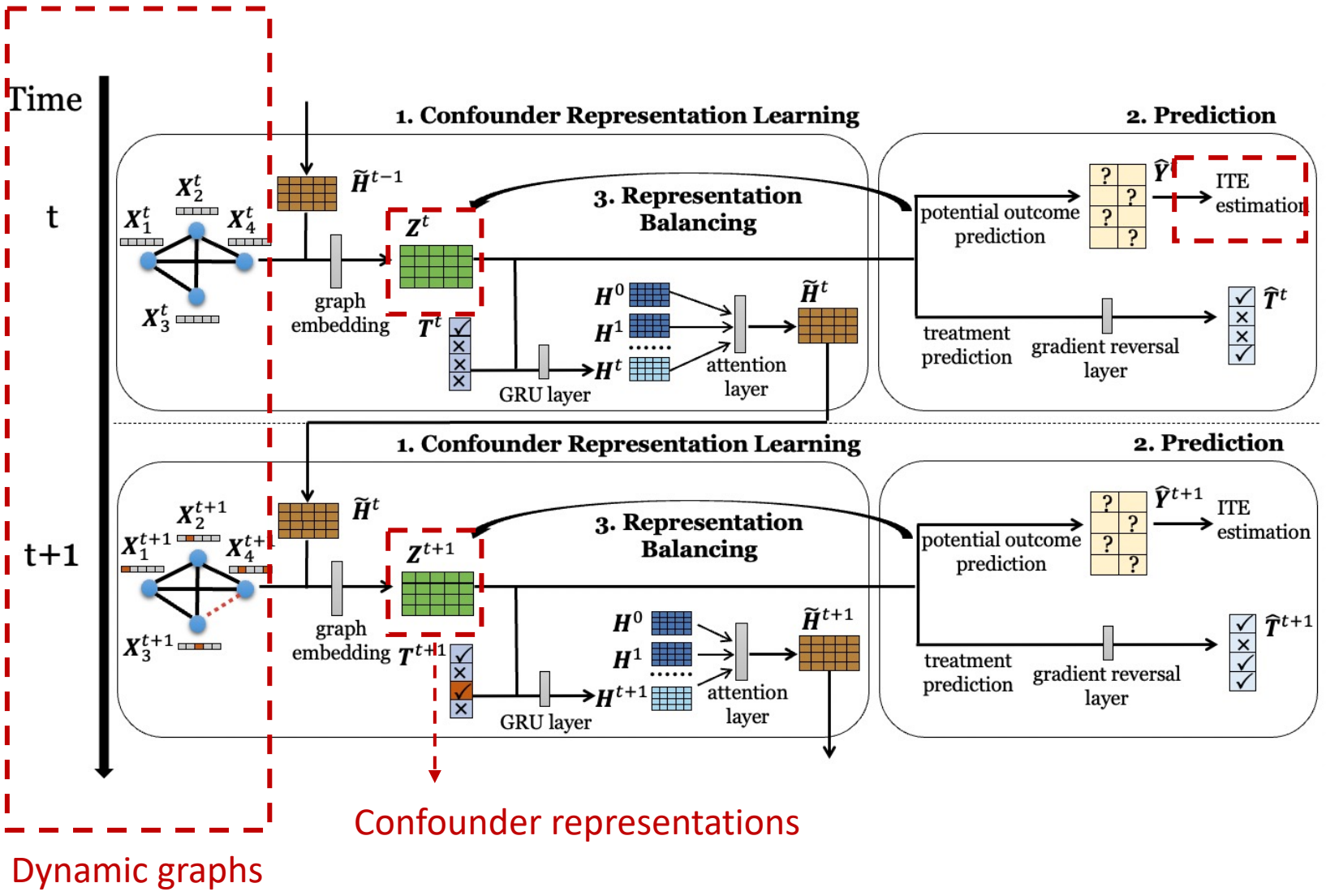
- Graph data can be proxy variables for hidden confounders
- Effective deep learning techniques are utilized
- Estimate ITE based on confounder representations

Problem Definition

- Given the time-evolving networked observational data $\{\boldsymbol{X}^t, \boldsymbol{A}^t, \boldsymbol{C}^t, \boldsymbol{Y}^t\}_1^T$ across T time stamps, the goal is to learn the ITE τ_i^t for each instance i at each time stamp t.



Proposed Method

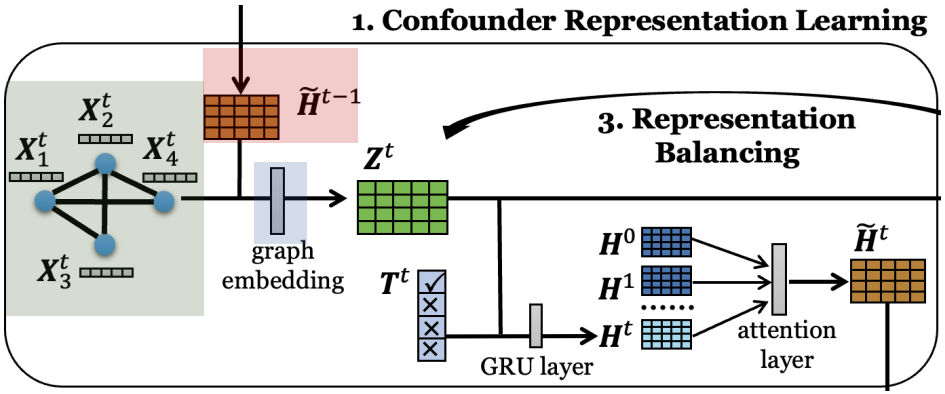


Proposed Method

- Confounder representation learning

Graph neural network | History embedding | Graph structure

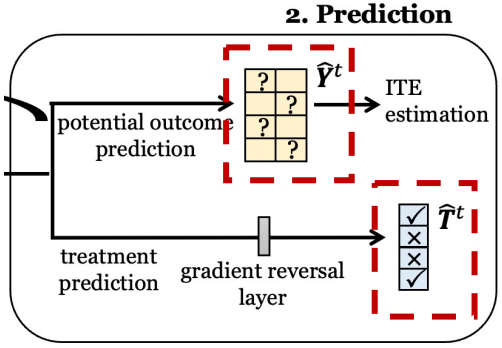
$$\boldsymbol{Z}_i^t = g(([\boldsymbol{X}^t, \widetilde{H}^{t-1}])_i, \boldsymbol{A}^t)$$



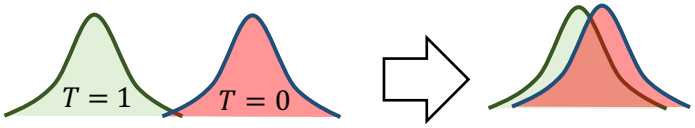
- Prediction for potential outcomes and treatment

$$\hat{Y}_{1,i}^t = f_1(\boldsymbol{Z}_i^t)$$
$$\hat{Y}_{0,i}^t = f_0(\boldsymbol{Z}_i^t)$$

ITE =



- Representation balancing
  - Help reduce the biases in ITE estimation



distribution of confounder representations

- Overall loss

$$\mathcal{L} = \begin{cases} \mathcal{L}_Y & \text{outcome prediction loss} \\ \mathcal{L}_T & \text{treatment prediction loss} \\ \mathcal{L}_B & \text{balancing loss} \end{cases}$$

Evaluation: Dataset

- Three real-world attributed networks

Dataset	Flickr	BlogCatalog	PeerRead
# of instances	7, 575	5, 196	6, 867 ~ 7, 601
# of links	236, 582	170, 626	11, 819
	~ 240, 374	~ 173, 524	~ 13, 684
# of features	12, 047	8, 189	1, 087
# of time stamps	25	20	17
ratio (%) of treated	48.72 ± 1.42	46.52 ± 1.58	49.51 ± 0.74
Avg ATE ± STD	14.35 ± 21.10	20.45 ± 16.63	56.25 ± 90.27

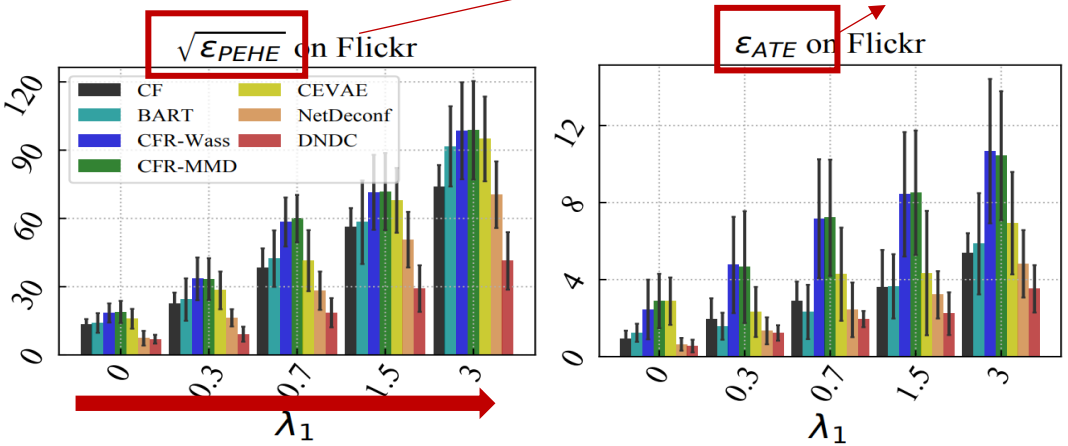
Evaluation

- Real-world social network Flickr (node=user, edge=friendship)
- Simulated causal problem (treatment, outcome, confounders)

The smaller, the better

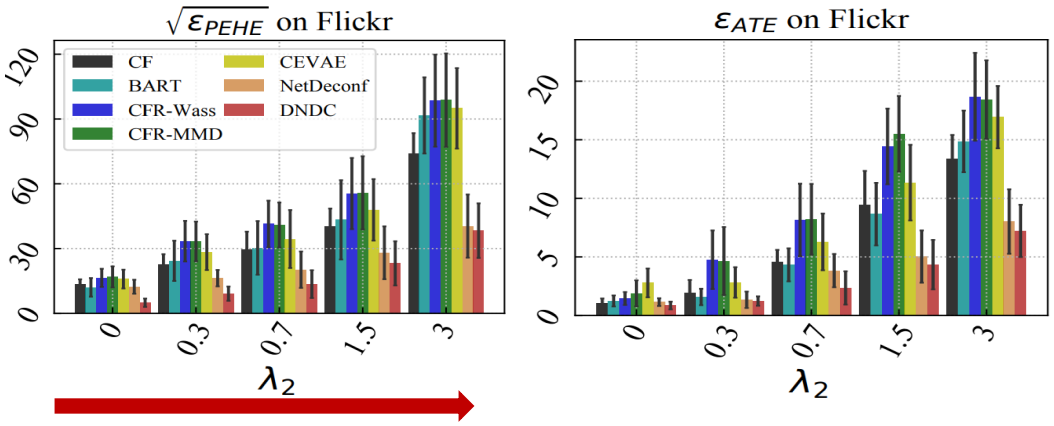
$$\sqrt{\epsilon_{PEHE}} = \sqrt{\frac{1}{n}\sum_{i\in[n]}(\tau_i - \hat{\tau}_i)^2}$$

$$\epsilon_{ATE} = |\frac{1}{n}\sum_{i\in[n]}\tau_i - \frac{1}{n}\sum_{i\in[n]}\hat{\tau}_i|$$



When $\lambda_1 \uparrow$, our method is better as it leverages historical information

λ_1 : the influence of historical information on confounders



When $\lambda_2 \uparrow$, our method is better as it leverages graph information

λ_2 : the influence of graph structure on confounders

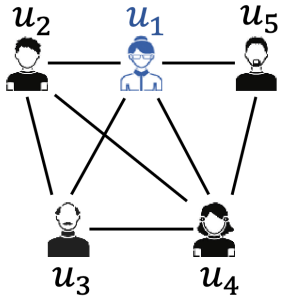
Outline

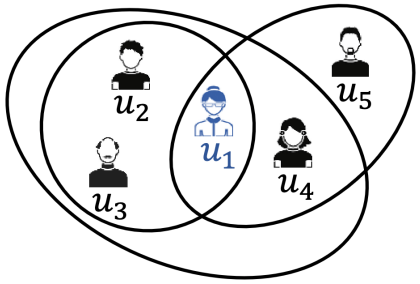
- Graph and Graph Neural Networks (GNNs)
- Deconfounding in Graphs
- Interference in Graphs

Hypergraph and High-order Interference

- In hypergraphs, each hyperedge can connect an arbitrary number of nodes, in contrast to an ordinary edge which connects exactly two nodes

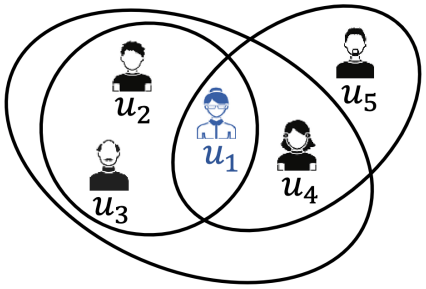
Ordinary graph

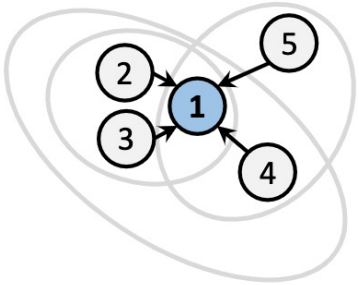




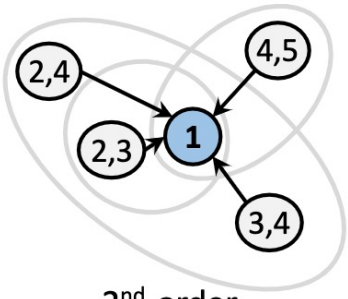
Hypergraph

- The interaction between u_2 and u_3 may also influence the exposure of the virus to u_1; i.e., $u_2 \times u_3 \rightarrow u_1$

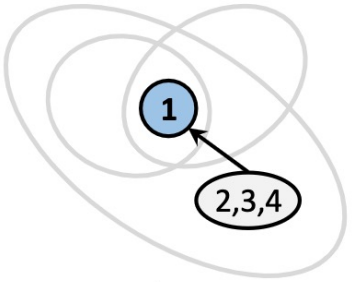




1st-order



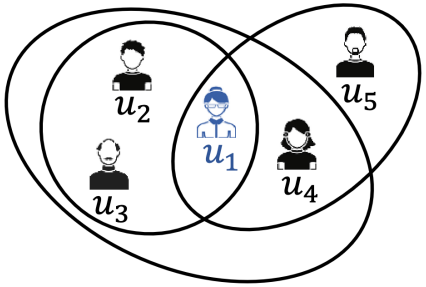
2nd-order



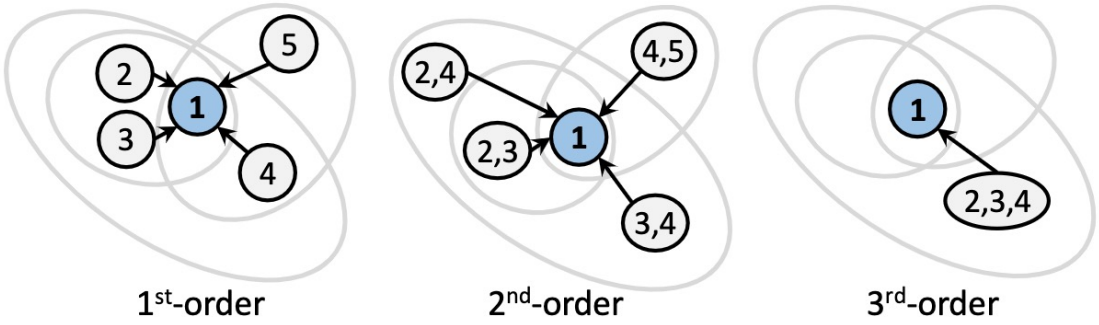
3rd-order

Causal Inference under Interference

- **Motivation**: High-order interference exists in hypergraphs



Hypergraph



(High-order) interference

- **Given**: observational data {***X***, ***H***, ***T***, ***Y***}, denoting features, hypergraph, treatment, and observed outcomes
- **Goal**: estimate the ITE for each individual (node) i:

$$\tau(\mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H}) = \mathbb{E}[Y_i^1 - Y_i^0 | X_i = \mathbf{x}_i, T_{-i} = \mathbf{T}_{-i}, X_{-i} = \mathbf{X}_{-i}, H = \mathbf{H}]$$
$$= \mathbb{E}[\Phi_Y(1, \mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H}) - \Phi_Y(0, \mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H})]$$

Potential outcome when T_i=1 or T_i=0

Treatment of other nodes

Assumptions

- **Assumption** 1. For any node i, given the node features X_i, the potential outcomes are independent with the treatment assignment and summary of neighbors

Assumption 1: (Unconfoundedness) no unobserved confounders exist

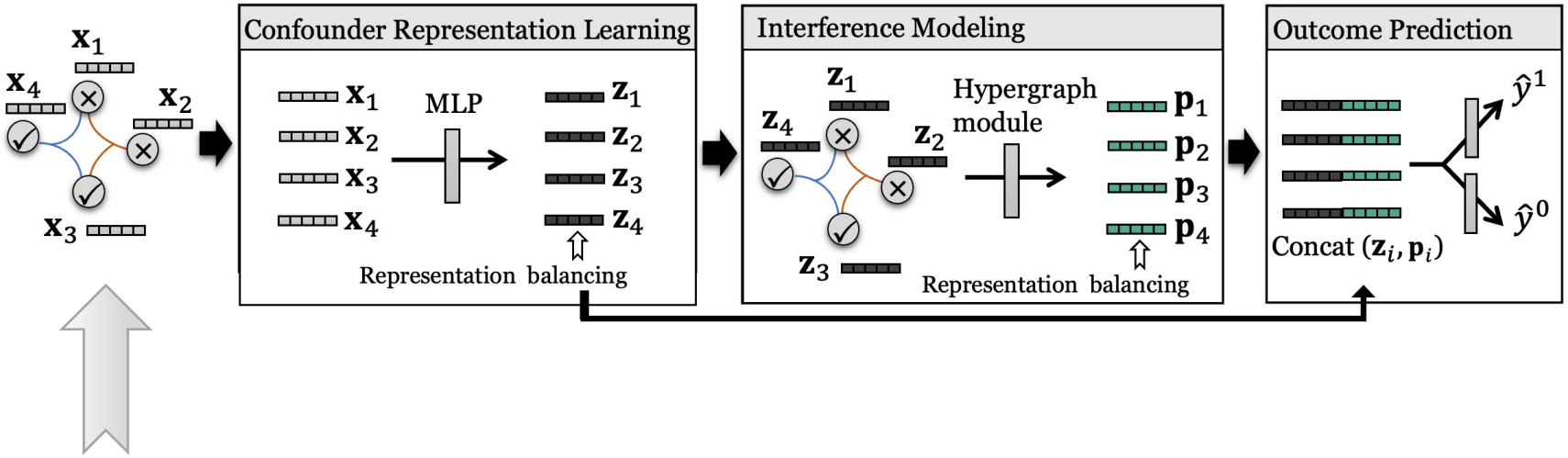
- **Assumption** 2. For any node i, any values of H, X_{-i}, and T_{-i}, if the output of a summary function $o_i = \mathrm{SMR}(H, T_{-i}, X_{-i})$ is determined, then the values of the potential outcomes with feature X_i are also determined

a function which characterizes all the "environmental" information related to node *i*.

Assumption 2: (Expressiveness of summary function)

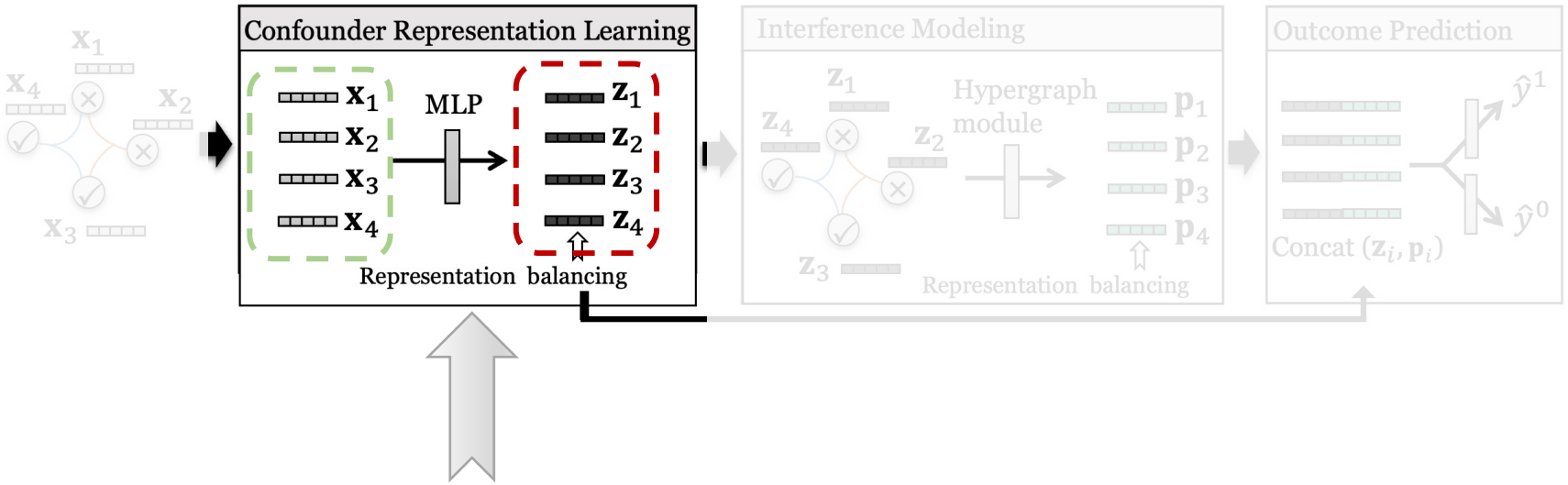
Theory (Identifiability): the defined ITE can be identifiable from observational data under the assumptions

Proposed Framework



Input hypergraph

Proposed Framework

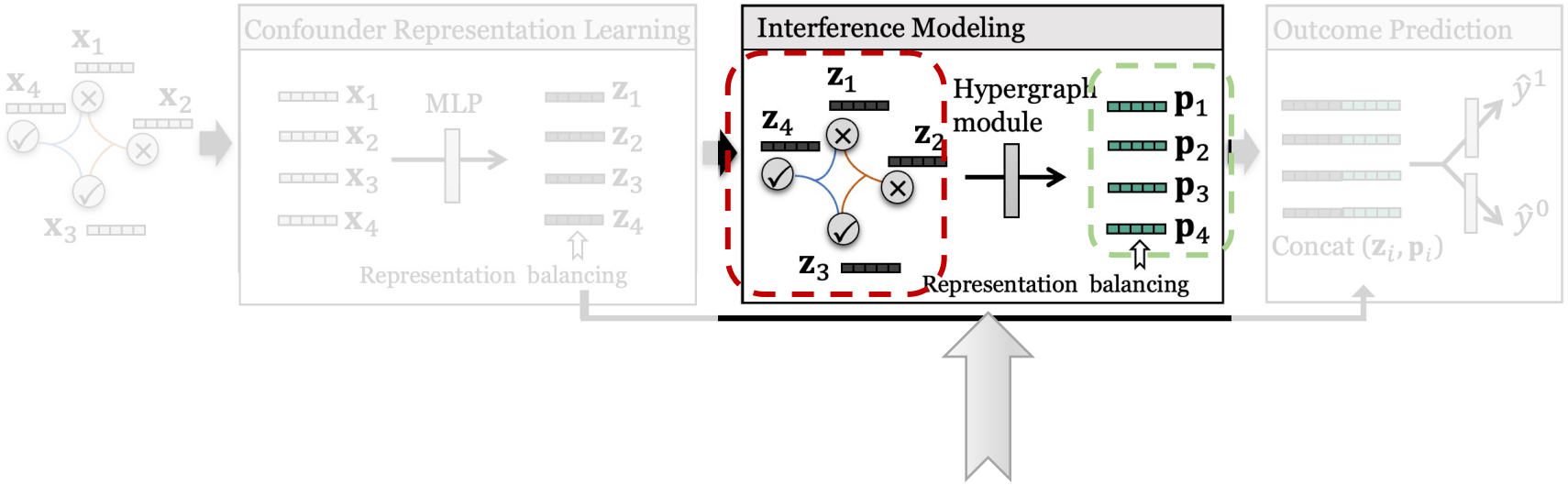


Confounder representation learning: encode the node features into a latent space to capture the confounders

$$\mathbf{z}_i = \mathrm{MLP}(\mathbf{x}_i)$$

Assumption 1 (unconfounedness)

Proposed Framework

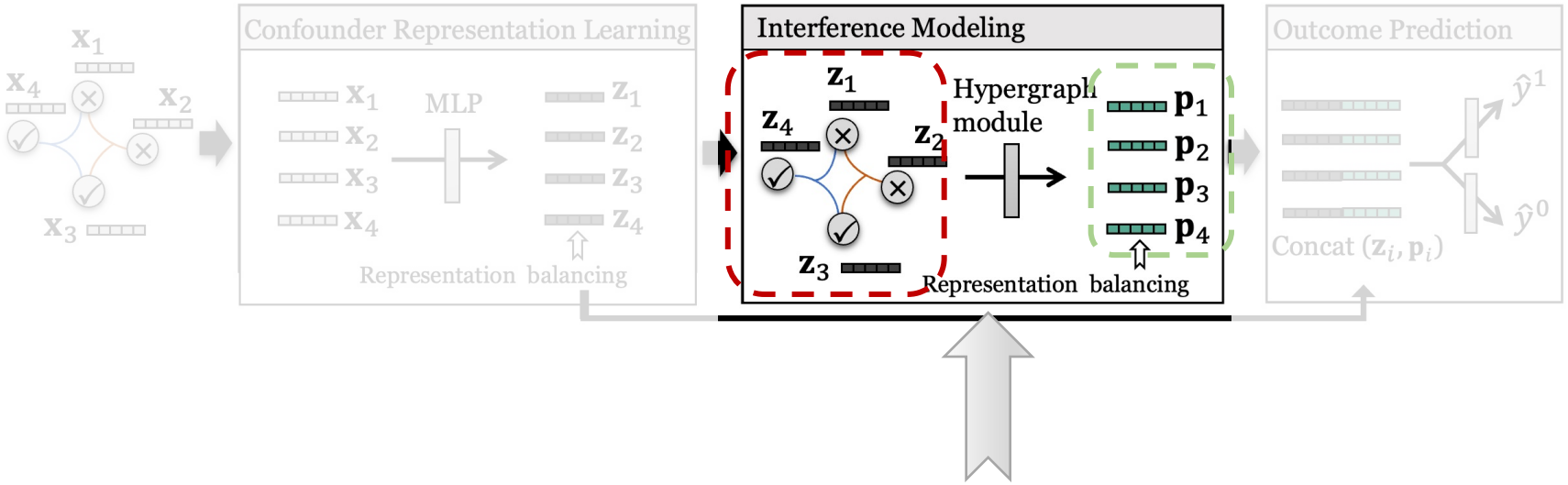


Interference Modeling: capture the high-order interference for each individual through representation learning

- Propagate the neighboring treatment assignment and confounder representations with a hypergraph module

Assumption 2 (Expressiveness of summary function)

Proposed Framework



Interference Modeling: capture the high-order interference for each individual through representation learning

- Hypergraph convoluntional network is applied in the hypergraph module:

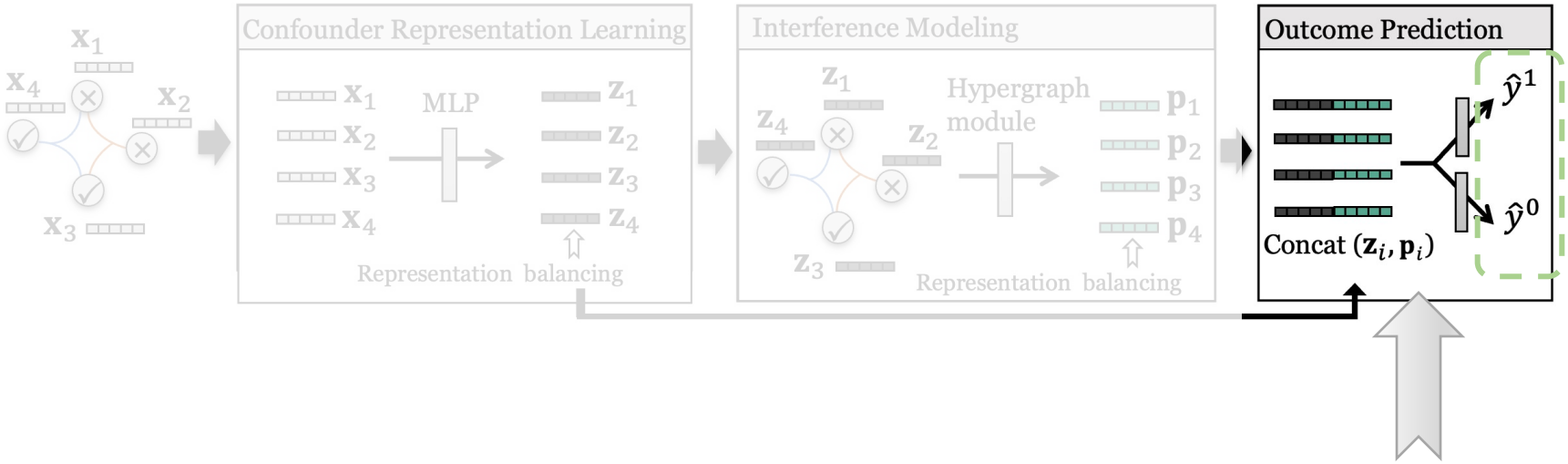
$$\mathbf{P}^{(l+1)} = \text{LeakyReLU}\left(\mathbf{L}\mathbf{P}^{(l)}\mathbf{W}^{(l+1)}\right)$$

Interference representation in $l+1$-th layer

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{H}\mathbf{B}^{-1}\mathbf{H}^{T}\mathbf{D}^{-1/2}$$

vanilla Laplacian matrix for the hypergraph

Proposed Framework



Outcome Prediction: predict the potential outcomes based on learned representations

$$\widehat{y}_i^1 = f_1([\mathbf{z}_i \| \mathbf{p}_i]),\ \widehat{y}_i^0 = f_0([\mathbf{z}_i \| \mathbf{p}_i])$$

$$\mathcal{L} = \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 + \alpha \mathcal{L}_b + \lambda \|\Theta\|^2$$

Model parameter regularization

Outcome prediction loss

Balancing loss

Evaluation

- **Dataset**: real-world contact hypergraph, node=person, hyperedge=physical contact
- **Simulation**: treatment=wear face mask, outcome=COVID-19 infection

Method	Linear		Quadratic	
	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}
LR	22.80 ±0.64	21.41± 0.74	414.17 ±3.94	192.80 ±2.97
CEVAE	19.36 ±0.80	8.63 ±0.78	315.01 ±2.53	188.47 ±4.27
CFR	25.23 ±0.01	18.28 ±0.02	392.56 ±4.33	189.75 ±4.80
Netdeconf	11.11 ±0.01	9.22 ±0.03	241.02 ±2.32	147.29 ±1.04
GNN-HSIC	9.38 ±0.44	6.91 ±0.38	114.28 ±3.62	81.21 ±2.53
GCN-HSIC	8.27 ±0.41	6.60 ±0.48	109.57 ±3.85	77.75 ±3.93
HyperSCI	**5.13 ±0.56**	**4.46 ±0.61**	**81.08 ±0.37**	**74.41 ±0.42**

Outcome simulation settings

The smaller, The better

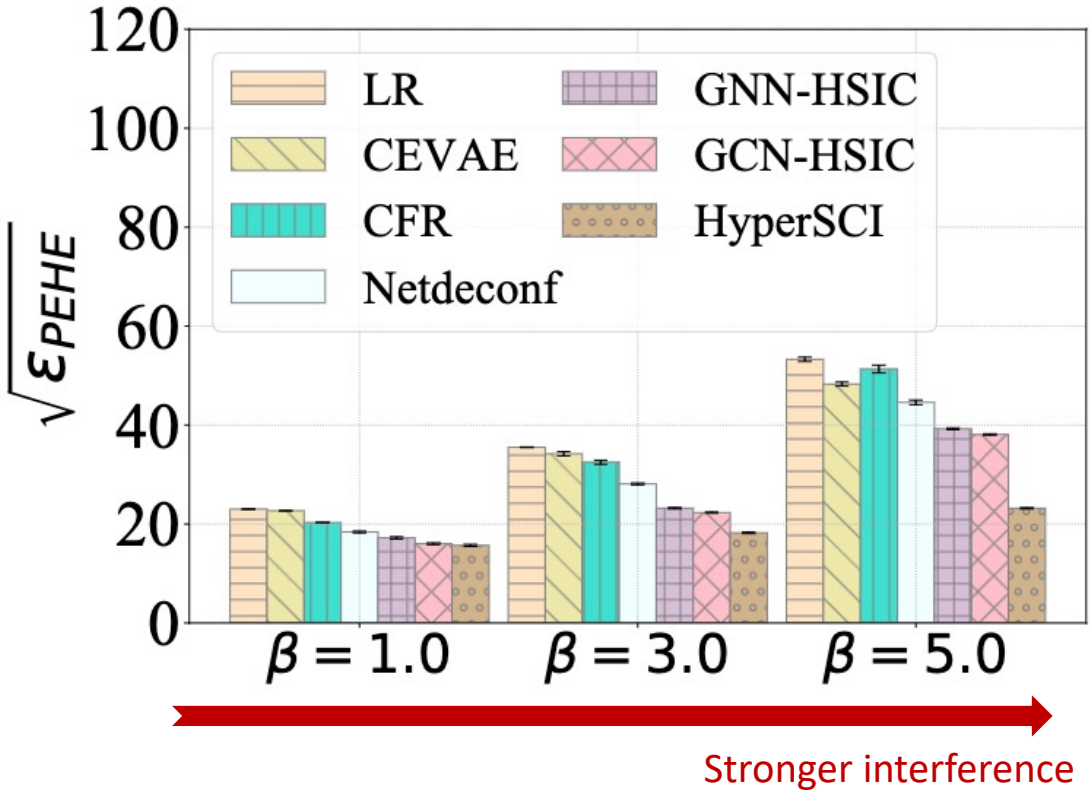
No graph

Projected graph

Our method

- Observations:
  - The proposed framework outperforms all the baselines under different outcome simulation settings
  - The high-order relational knowledge can help causal effect estimation

Evaluation



- Main observations:
  - **Interference**. Our framework outperforms all the baselines, especially under higher interference (larger β)

References

- Related work:
  - Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.
  - Velickovic P, Cucurull G, Casanova A, et al. Graph attention networks[J]. stat, 2017, 1050(20): 10-48550.
  - Guo R, Li J, Liu H. Learning individual causal effects from networked observational data[C]//Proceedings of the 13th international conference on web search and data mining. 2020: 232-240.
  - Ma J, Guo R, Chen C, et al. Deconfounding with networked observational data in a dynamic environment[C]//Proceedings of the 14th ACM International Conference on Web Search and Data Mining. 2021: 166-174.
  - Ma J, Wan M, Yang L, et al. Learning causal effects on hypergraphs[C]//Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022: 1202-1212.
- Some slides are from:
  - Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound. Lecture 18. The University of British Columbia.

Reading Materials

- Guo R, Li J, Liu H. Learning individual causal effects from networked observational data[C]//WSDM. 2020. https://dl.acm.org/doi/pdf/10.1145/3336191.3371816
- Ma J, Guo R, Chen C, et al. Deconfounding with networked observational data in a dynamic environment[C]//WSDM. 2021. https://dl.acm.org/doi/pdf/10.1145/3437963.3441818?casa_token=mZOj84z1eAwAAAAA:dRvHC8aRFc801QJpt1vh2Xs1BHKQn1MxiOJ4zNQFcKP4ovd_dOwLCOX9KBKLjCtL4dMI6saRSys2mA
- Ma J, Wan M, Yang L, et al. Learning causal effects on hypergraphs[C]//ACM SIGKDD. 2022. https://dl.acm.org/doi/pdf/10.1145/3534678.3539299

Thank you!
Questions?