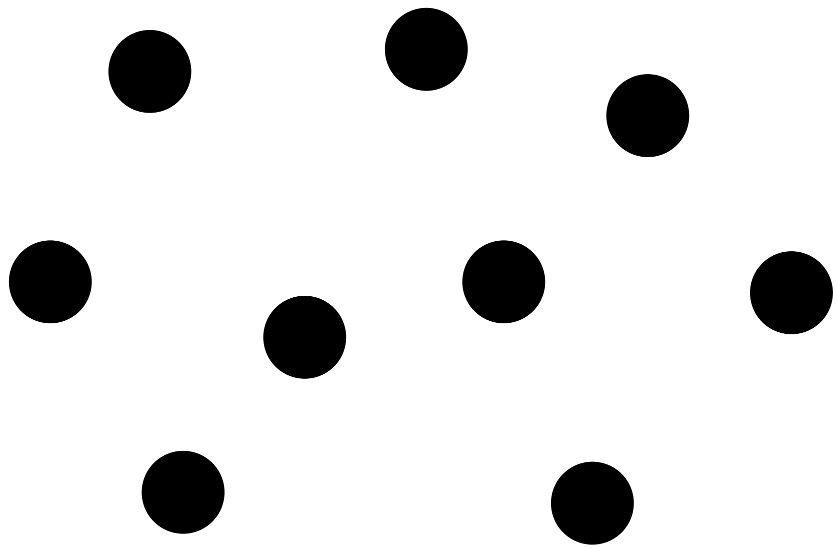
Network Generation by Deep Reinforcement Learning

Yang Yang, Jiarong Xu
Zhejiang University

Why Networks?

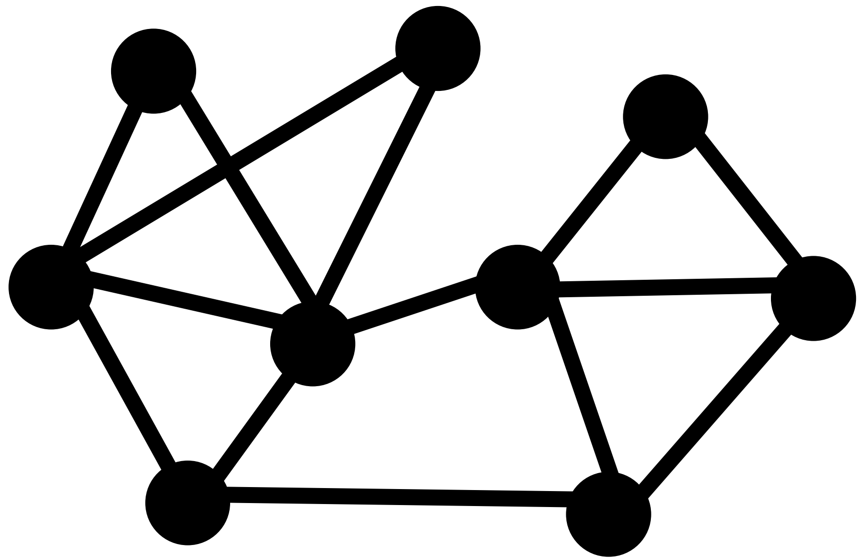
- Networks are a natural language for describing and modeling complex systems.



Independent Data

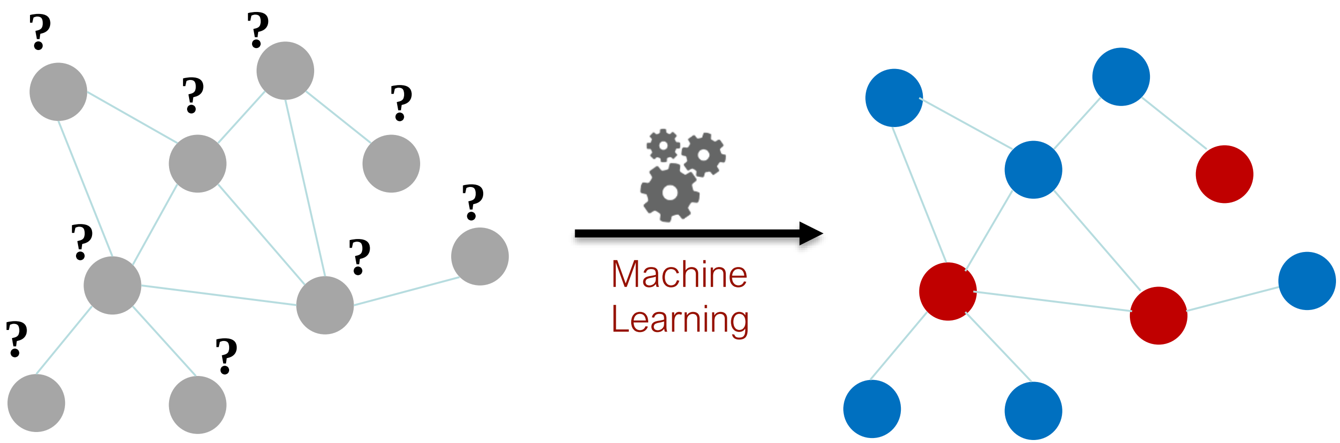
Why Networks?

- Networks are a natural language for describing and modeling complex systems.



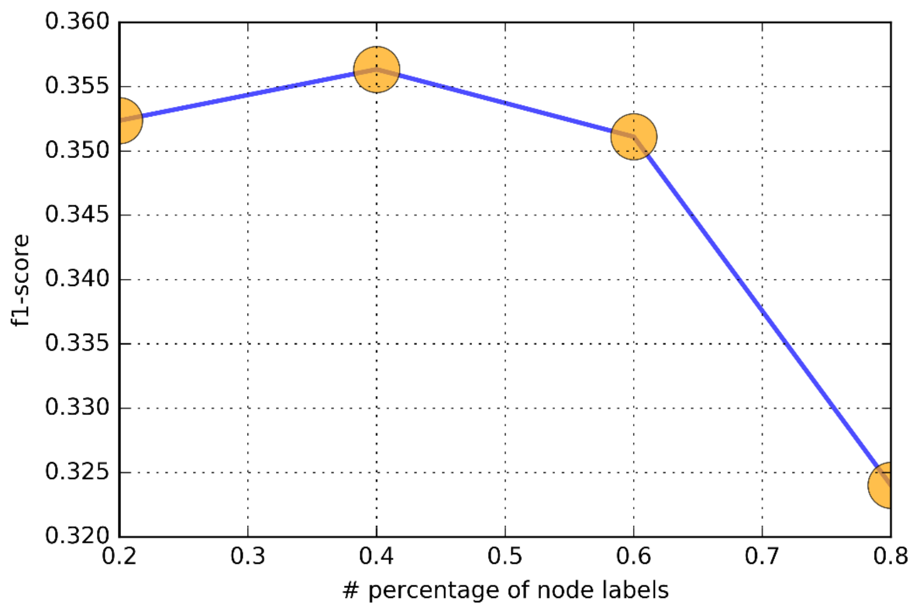
Network!

Application: Vertex Classification



Noisy Network Data

- In real world, network data is **hard to obtain** and consists of **noises**
  - Noises are caused by **incomplete and biased data sampling**, **human subjectivity**, and **inconsistent with the downstream task**.

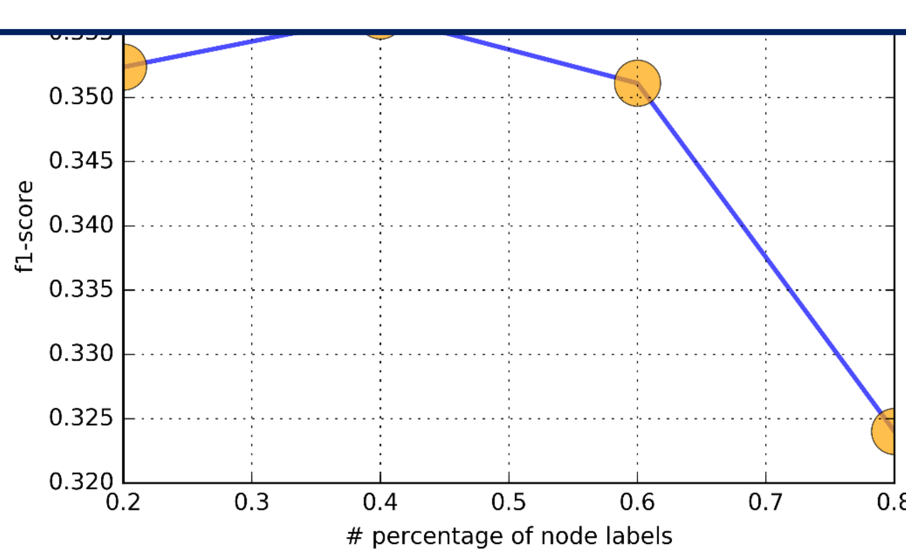


Task performance drops as more node labels are obtained!

Noisy Network Data

- In real world, network data is **hard to obtain** and consists of **noises**
  - Noises are caused by **incomplete and biased data sampling**, **human subjectivity**, and **inconsistent w**

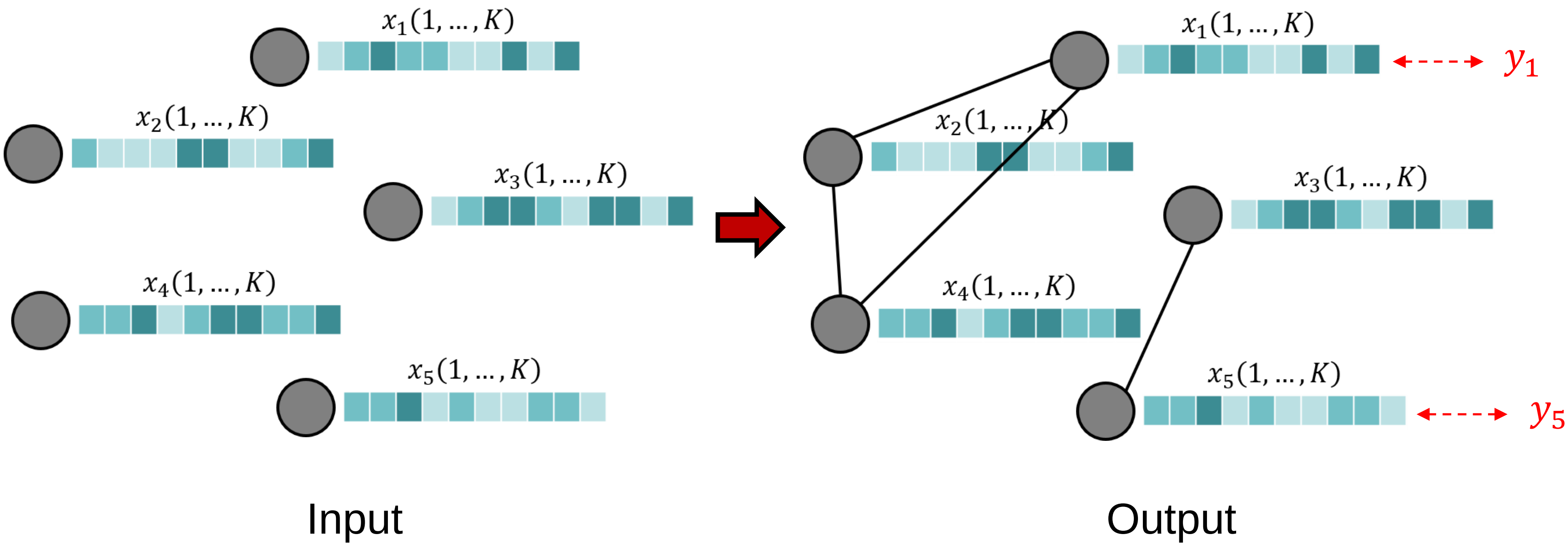
How to construct a reliable network?



Task performance drops as more node labels are obtained!

Task

- Construct network by considering **node features** and optimizing **task performance**.

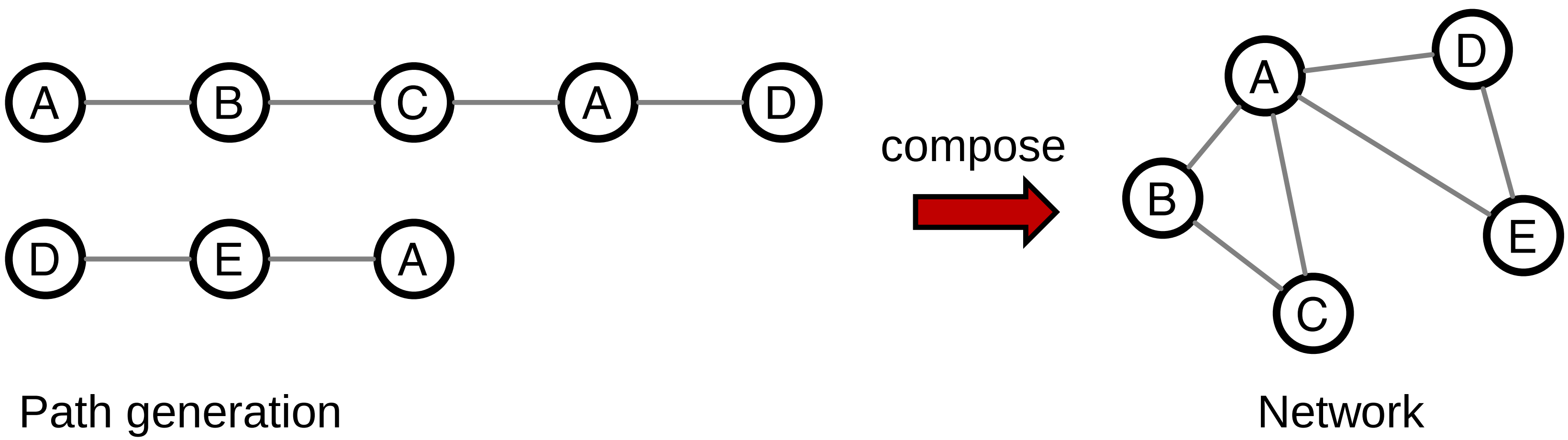


Related Work

- **Link prediction: given two nodes, determine if there exist an edge.**
  - Hard to preserve macro-level network properties.
- **Network generation model: design a certain mechanism to generate (nodes and) edges.**
  - Requires rich domain knowledge and hard to generalize.

General Idea

- A network is consisted by several **paths** among nodes.



Path generation can be formulated as Markov Decision Process!

Our Approach

- Reinforcement learning framework
  - **State:** $(v_0, \ldots, v_t)$, where v_t is the current node
  - **Action:** v_{t+1} , create an edge between v_t and v_{t+1}

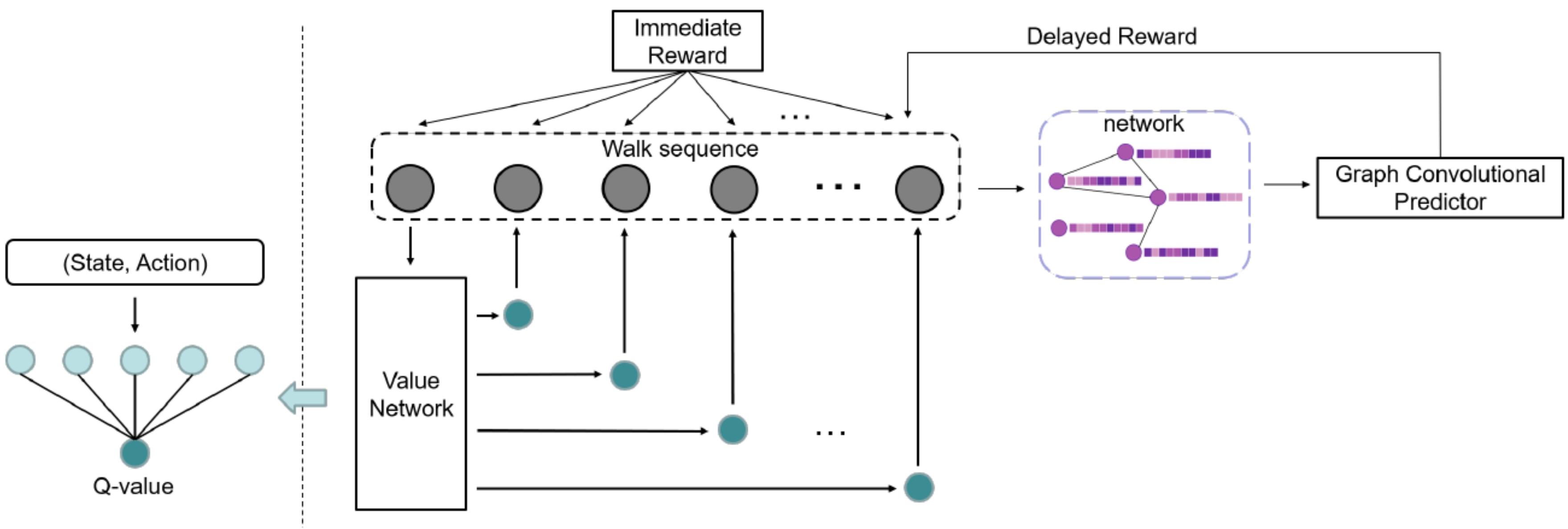


Figure 1: An overview of the proposed network reconstruction method.

Our Approach

- Reinforcement learning framework
  - **Reward**:

➢ **Immediate reward**: a given network (as train data) guided

$$r_i(v_{t+1}|v_t) = \begin{cases} 0, (v_t, v_{t+1}) \notin E \text{ } or \text{ } (v_t, v_{t+1}) \in \big((v_0, v_1), (v_1, v_2), \dots, (v_{t-1}, v_t)\big) \\ 1, (v_t, v_{t+1}) \in E \end{cases}$$

edges set

➢ **Delayed reward**: task guided

$$r_d\big(v_T|(v_0, v_1, \dots, v_T)\big) = GCN\big(G\_net\big((v_0, v_1, \dots, v_T)\big), features\big)$$

Learning Algorithm

Algorithm 2 Overall Training Process

Initialize positive replay memory $\mathcal{D}_p$ to capacity N
Initialize negative replay memory $\mathcal{D}_n$ to capacity M
Initialize action-value function Q with random weights
repeat
  Initialize sequence $s_0 = (v_0)$
  for $t = 0$ **to** T **do**
    With probability ε select a random action a_t
    Otherwise select $a_t = \max_a Q^*(v_t, a; \theta)$
    Execute action a_t in emulator and observe reward r_t and next node v_{t+1}
    Set $s_{t+1} = (v_0, \cdots, v_{t+1})$
    if $r_t > 0$ **then**
      Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_p$
    else
      Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_n$
    end if
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}_p$ and $\mathcal{D}_n$ with a proportion of 1:4
    Set $y_j = \begin{cases} GCN\left(G\_net\left(s_{j+1}\right), features\right) & len\left(s_{j+1}\right) == T \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{others} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(s_j, a_j; \theta)\right)^2$
  end for
until *episodes* finish

Select & execute actions with e-greedy

Construct experience replay

Train Q network

Experimental Setup

- Dataset:
  - Terrorist network:
    - Terrorists(1206) vs. Non-terrorists(2767)
    - Each node has 45 features
  - Citation network: Cora & Pubmed
    - Verties: documents
    - Edges: citation links between documents
    - Features: sparse bag-of-words representation

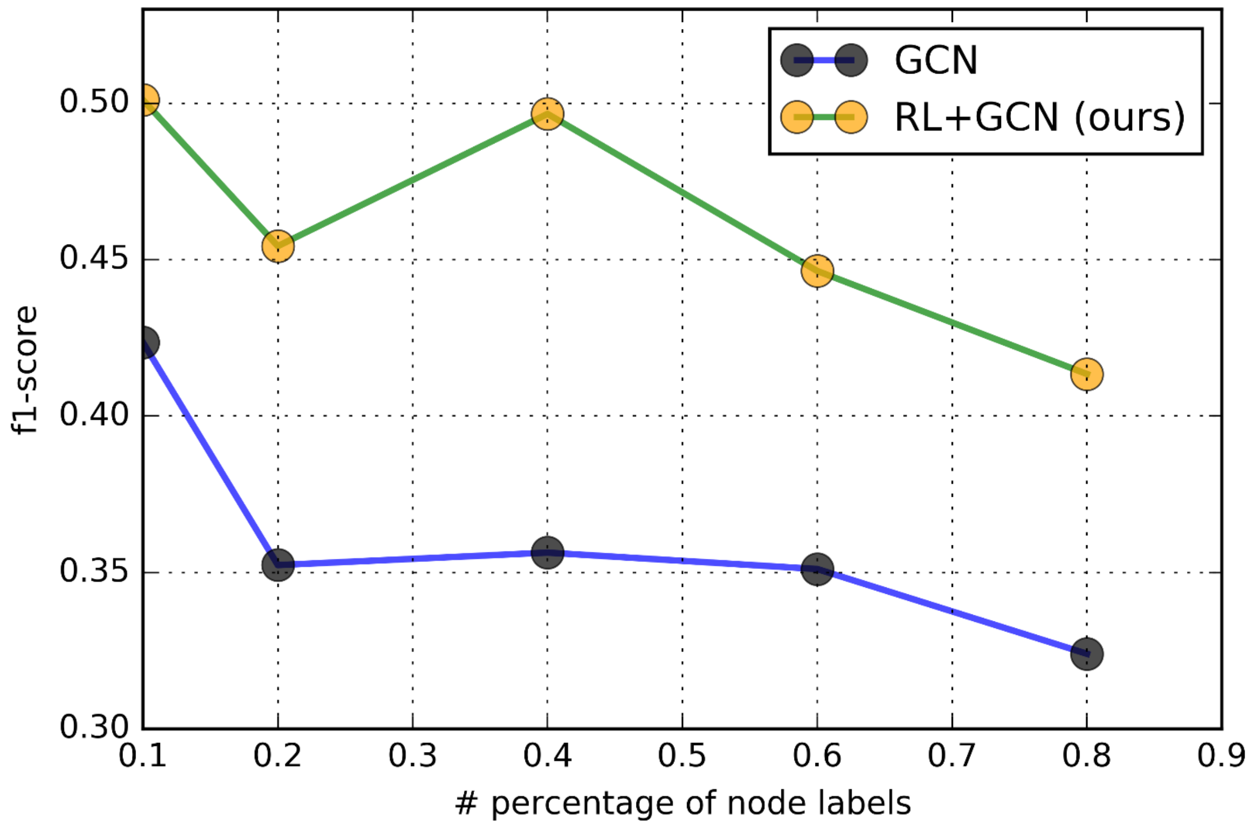
DATASET	TYPE	#VERTEIES	#EDGES	#CLASSES	#FEATURES
Hikvision	Terrorist Network	3,973	7,600	2	45
Cora	Citation network	2,708	5,429	7	1,433
Pubmed	Citation network	19,717	44,338	3	500

- Downstream task: **node classification**

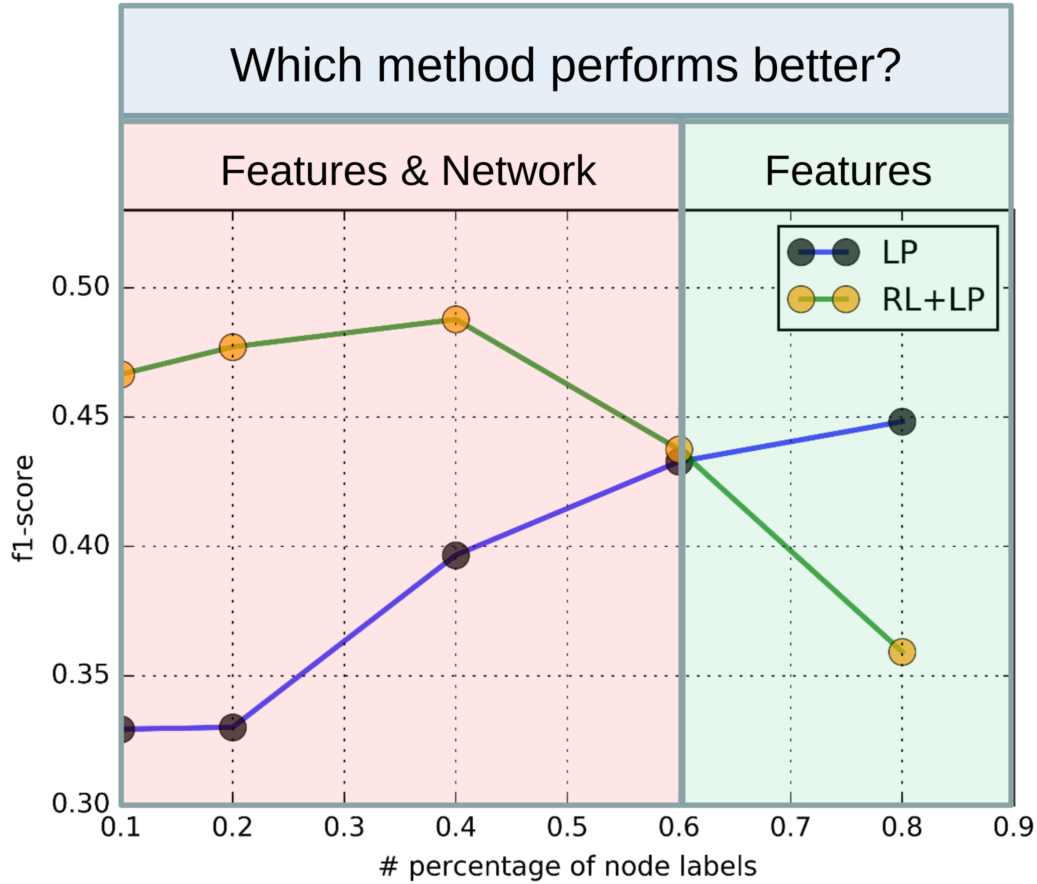
Experimental Setup

- Baselines:
  - **MLP**: Conduct MLP classification based on features
  - **LP**: Construct similarity matrix based on features, and conduct label propagation on it.
  - **RL + LP**: Extract similarity matrix from RL, and conduct label propagation on it.
  - **GCN**: Conduct GCN classification based on pre-built network and features.
  - **RL + GCN**: Reconstruct network from RL, and conduct GCN classification based on the reconstructed network and features.

RL Does Help!


GCN
RL+GCN (ours)
0.50
0.45
0.40
0.35
0.30
f1-score
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
percentage of node labels

Experimental Results – Terrorist Network



With the percentage of node labels **decreases**:

- Method based on **features** gets **worse**
- Method based on **both features and network** gets **better**

Experimental Results

			MLP	LP	RL+LP	GCN	RL+GCN
# percentage of node labels	0.8	Accuracy	0.82087	0.75536	0.37872	0.69560	0.42893
		Precision	0.73687	0.70622	0.28969	0.39456	0.42893
		Recall	0.56500	0.32961	0.28969	0.27488	0.75829
		F1-score	0.63910	0.44826	0.35928	0.32402	0.41344
	0.6	Accuracy	0.81544	0.73868	0.29811	0.68616	0.43837
		Precision	0.71940	0.67969	0.29198	0.40663	0.30612
		Recall	0.55546	0.31843	0.87212	0.30892	0.82380
		F1-score	0.62651	0.43276	0.43748	0.35111	0.44637
	0.4	Accuracy	0.81112	0.72945	0.32257	0.68330	0.59941
		Precision	0.70925	0.70667	0.32257	0.42828	0.38861
		Recall	0.55789	0.27568	1.00000	0.30511	0.68759
		F1-score	0.62407	0.39663	0.48779	0.35635	0.49657
	0.2	Accuracy	0.80441	0.69991	0.31331	0.66467	0.39321
		Precision	0.69573	0.54907	0.31331	0.40903	0.30908
		Recall	0.55748	0.23594	1.00000	0.30950	0.85699
		F1-score	0.61838	0.33006	0.47713	0.35237	0.45432
	0.1	Accuracy	0.80033	0.70274	0.30425	0.66303	0.52657
		Precision	0.68978	0.52515	0.30425	0.41698	0.36654
		Recall	0.55646	0.23989	1.00000	0.30633	0.79143
		F1-score	0.61545	0.32934	0.46655	0.35319	0.50103

Experimental Results – Terrorist Network

			MLP	LP	RL+LP	GCN	RL+GCN
# percentage of node labels	0.8	Accuracy	0.82087	0.75536	0.37872	0.69560	0.42893
		Precision	0.73687	0.70622	0.28969	0.39456	0.42893
		Recall	0.56500	0.32961	0.28969	0.27488	0.75829
		F1-score	0.63910	0.44826	0.35928	0.32402	0.41344
	0.6	Accuracy	0.81544	0.73868	0.29811	0.68616	0.43837
		Precision	0.71940	0.67969	0.29198	0.40663	0.30612
		Recall	0.55546	0.31843	0.87212	0.30892	0.82380
		F1-score	0.62651	0.43276	0.43748	0.35111	0.44637
	0.4	Accuracy	0.81112	0.72945	0.32257	0.68330	0.59941
		Precision	0.70925	0.70667	0.32257	0.42828	0.38861
		Recall	0.55789	0.27568	1.00000	0.30511	0.68759
		F1-score	0.62407	0.39663	0.48779	0.35635	0.49657
	0.2	Accuracy	0.80441	0.69991	0.31331	0.66467	0.39321
		Precision	0.69573	0.54907	0.31331	0.40903	0.30908
		Recall	0.55748	0.23594	1.00000	0.30950	0.85699
		F1-score	0.61838	0.33006	0.47713	0.35237	0.45432
	0.1	Accuracy	0.80033	0.70274	0.30425	0.66303	0.52657
		Precision	0.68978	0.52515	0.30425	0.41698	0.36654
		Recall	0.55646	0.23989	1.00000	0.30633	0.79143
		F1-score	0.61545	0.32934	0.46655	0.35319	0.50103

?

However, the network information hurts the performance overall… Why?

Labels Sensitivity

Table 3: Classification performance on noisy citation network (add 60% noises) with different ratios of vertex labels

		80%		60%		40%	
		micro-f1	macro-f1	micro-f1	macro-f1	micro-f1	macro-f1
Noisy Cora	LP	0.16236	0.05799	0.12177	0.04052	0.12308	0.04322
	RL+LP	0.42620	0.42620	0.13284	0.03350	0.14277	0.05799
	GCN	0.61808	0.48967	0.60526	0.48245	0.60283	0.48175
	RL+GCN (ours)	**0.76384**	**0.71929**	**0.73155**	**0.67555**	**0.72246**	**0.67023**
Noisy Pubmed	LP	0.34559	0.32646	0.38177	0.27309	0.39887	0.29663
	RL+LP	0.38565	0.18554	0.39318	0.20926	0.39363	0.24846
	GCN	0.71661	0.70437	0.71991	0.70622	0.72292	0.70830
	RL+GCN (ours)	**0.86105**	**0.86086**	**0.86332**	**0.86308**	**0.85775**	**0.85739**

		20%		10%	
		micro-f1	macro-f1	micro-f1	macro-f1
Noisy Cora	LP	0.15136	0.03997	0.15135	0.04649
	RL+LP	0.12921	0.03269	0.13002	0.03288
	GCN	0.61343	0.54758	0.60008	0.56347
	RL+GCN (ours)	**0.70051**	**0.65302**	**0.66571**	**0.62909**
Noisy Pubmed	LP	0.39952	0.19033	0.35405	0.24257
	RL+LP	0.39565	0.26209	0.38905	0.18680
	GCN	0.72291	0.71060	0.70830	0.70830
	RL+GCN (ours)	**0.85280**	**0.85251**	**0.84419**	**0.84352**

Noises Sensitivity

Table 4: Classification performance on citation network (given 80% vertex labels) with different ratios of noises

		20% noises		40% noises		60% noises		80% noises	
		micro-f1	macro-f1	micro-f1	macro-f1	micro-f1	macro-f1	micro-f1	macro-f1
Cora	GCN	**0.79612**	**0.74578**	0.71089	0.61443	0.61808	0.48967	0.47343	0.33818
	RL+GCN (ours)	0.78044	0.744	**0.76753**	**0.71932**	**0.76384**	**0.71929**	**0.78044**	**0.74580**
Pubmed	GCN	0.71660	0.70437	0.77350	0.76704	0.71661	0.70437	0.67135	0.65223
	RL+GCN (ours)	**0.86055**	**0.86031**	**0.85928**	**0.85959**	**0.85928**	**0.85941**	**0.85953**	**0.85972**

Summary

- We study the problem of **network reconstruction** in a **deep reinforcement learning framework**.
- Our method efficiently improve the downstream task performance, comparing with other method utilizing network information.
- Issues remain:
  - MLP only based on features performs best?
  - Do the generated edges have a clear physical meaning?
  - **We need to dig deeper on the data!**

Thank you!

Q&A