中国人民大学 RENMIN UNIVERSITY OF CHINA
THE UNIVERSITY OF QUEENSLAND AUSTRALIA

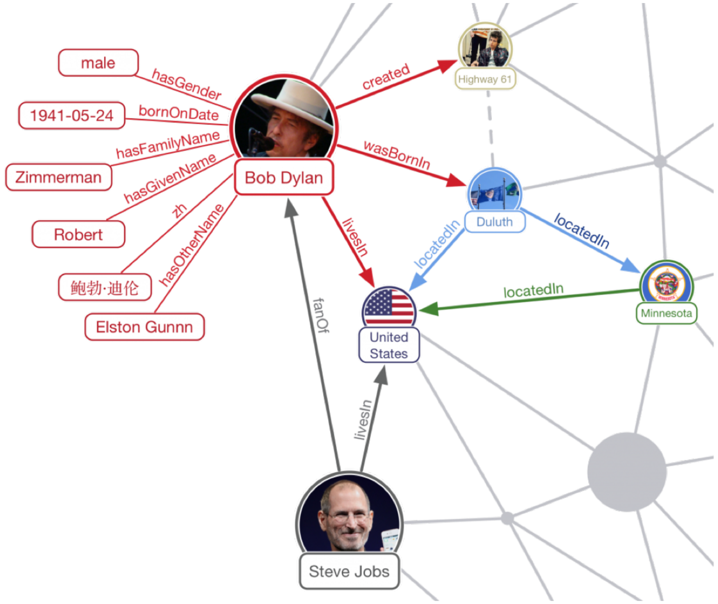
Graph Self-supervised Learning for Anomaly Detection and Recommendation

Presented by Jing Zhang (RUC)

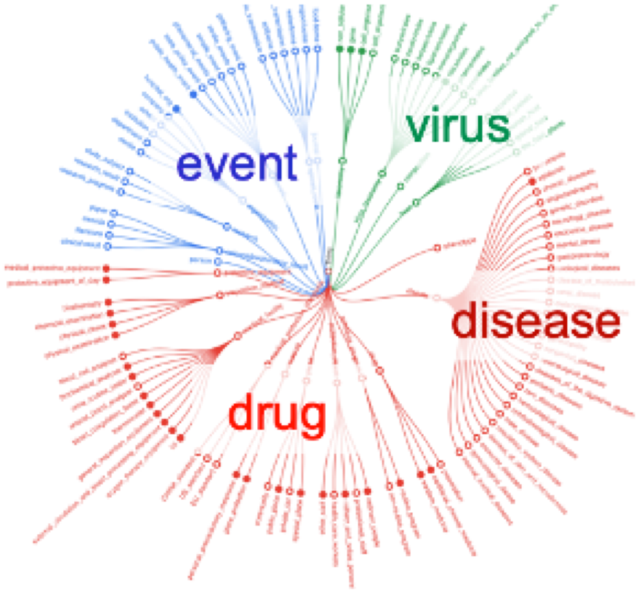
Collaborated with Yanling Wang (RUC), Bowen Hao (RUC), Hongzhi Yin (UQ), Shasha Guo (RUC), Cuiping Li (RUC) and Hong Chen (RUC)

Networked Data

Social Network

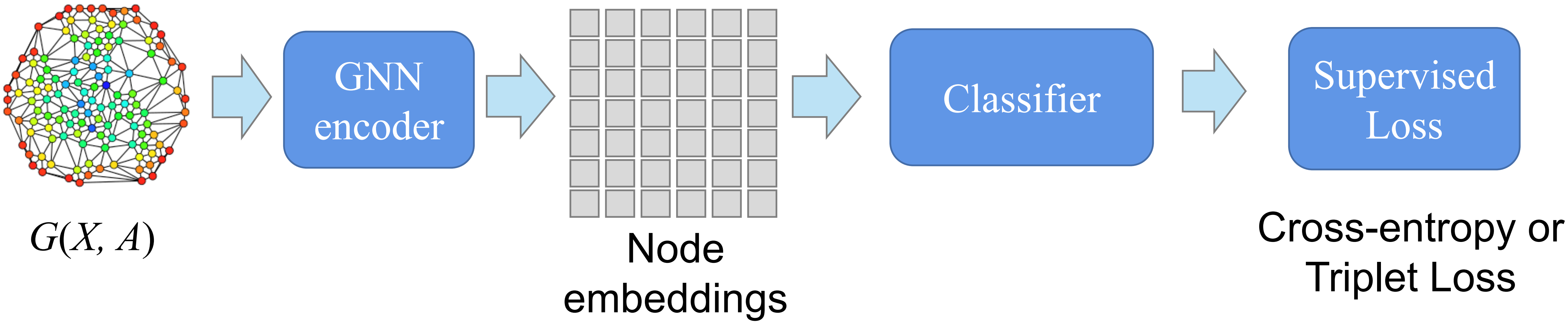


Knowledge Graph



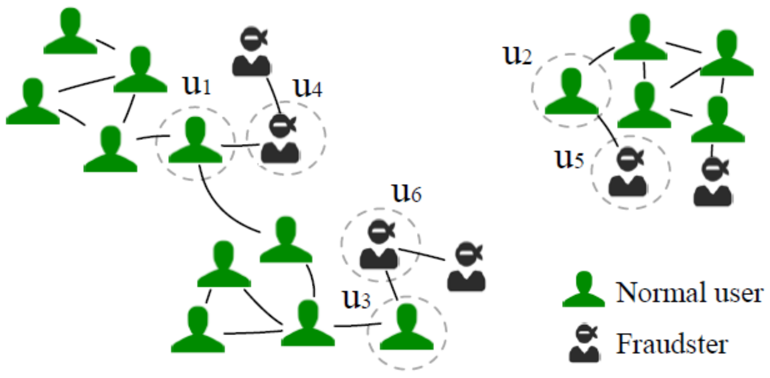
COVID Graph

Graph Neural Network



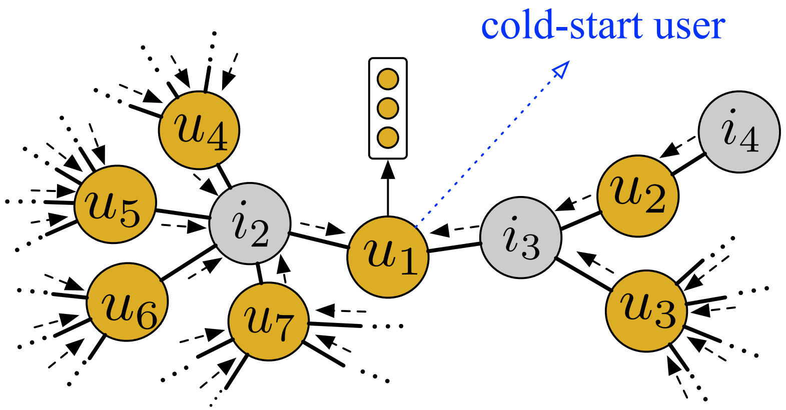
End-to-End Training Suffers from Limitations

- Data Inconsistency
  - The node patterns and the label semantics disagree with each other.
  - Dilemma: to learn the intrinsic graph properties or to capture the label semantics?



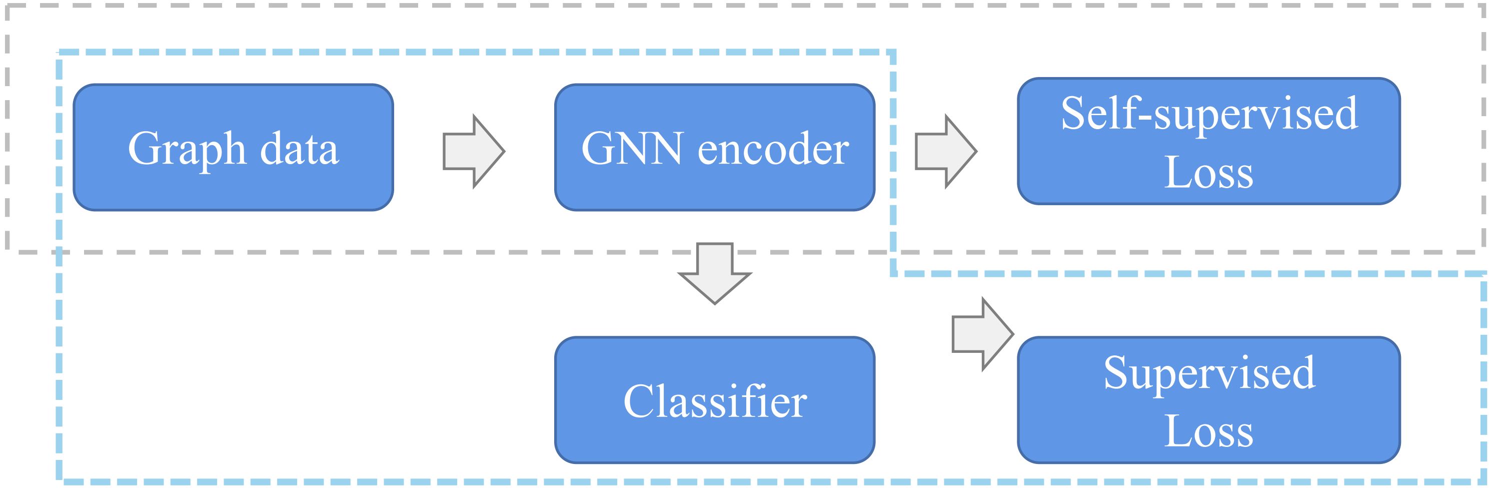
Anomaly Detection

- Sparse Interactions
  - The nodes have sparse interactions with others.
  - Prevent learning high-quality embeddings.



Recommendation

Two-stage Training via Self-supervised Learning

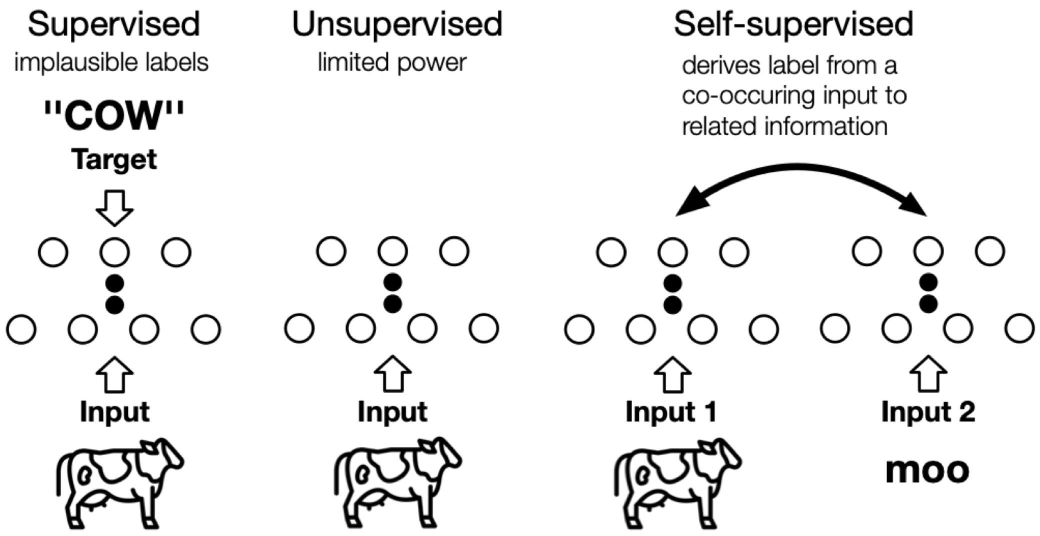


Stage 1:
Self-supervised learning (SSL) of encoder

Stage 2:
Supervised learning of encoder and classifier

- Motivation
  - Due to the limited power of label information, we alternate to train the feature encoder via the self-supervision.
  - Yann LeCun: "self-supervised learning is the cake, supervised learning is the icing on the cake, reinforcement learning is the cherry on the cake".

What is Self-supervised Learning



- Reaches higher accuracy with fewer labels and plateaus to the same performance as the supervised baseline.
- Is more robust and stable.
- Outperforms supervised distribution in out-of-distribution detection on difficult, near-distribution outliers.

Image from Liu et al. 2021. Self-supervised Learning: Generative or Contrastive. TKDE

Graph Self-supervised Learning

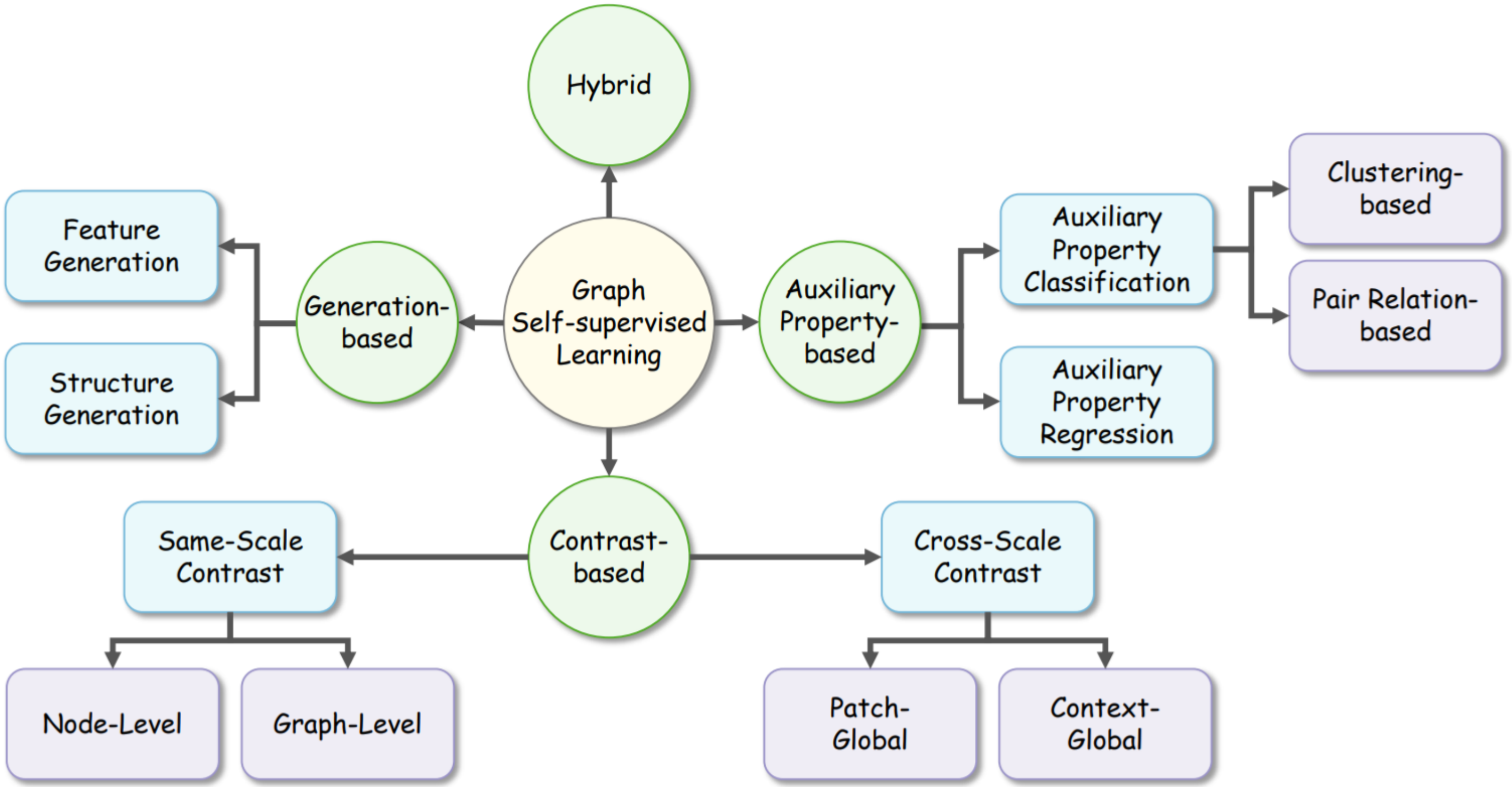
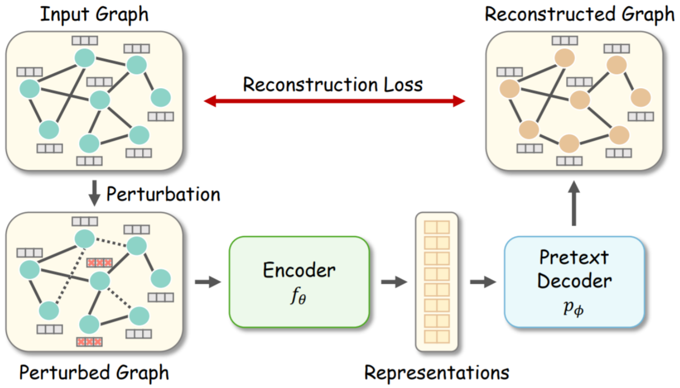


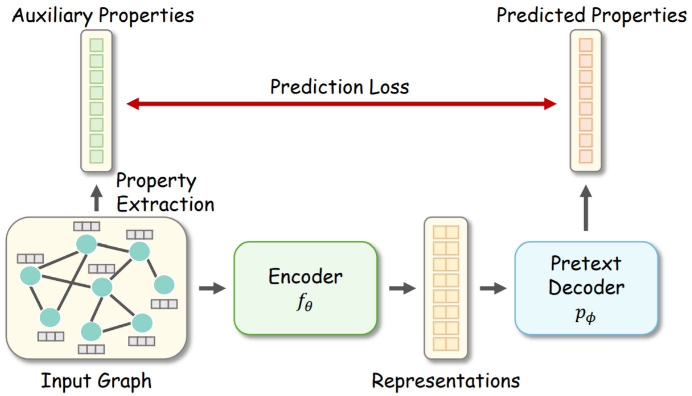
Image from Liu et al. 2021. Self-supervised Learning: Generative or Contrastive. CoRR abs/2103.00111 (2021)

Graph Self-supervised Learning

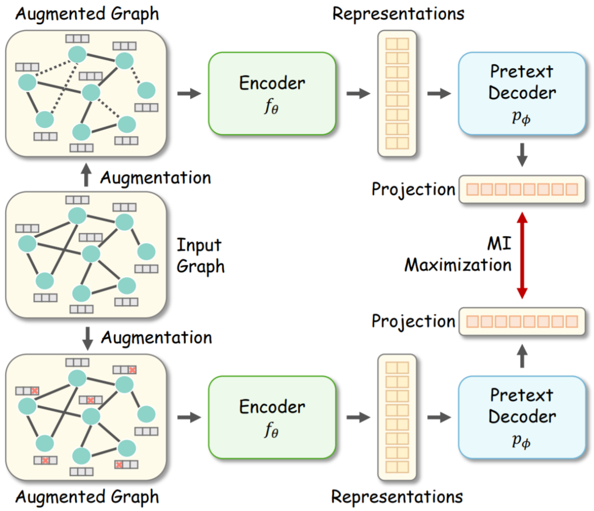
Generation-based



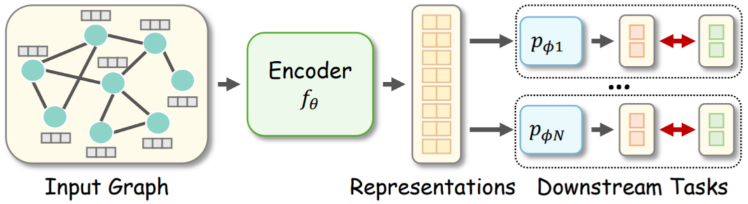
Auxiliary property-based



Contrast-based



Hybrid



Images from Liu et al. 2021. Self-supervised Learning: Generative or Contrastive. CoRR abs/2103.00111 (2021)

Graph Self-supervised Learning

- Can all the graph SSL methods benefit the downstream tasks?
- How to devise the graph SSL objective to improve the downstream tasks?

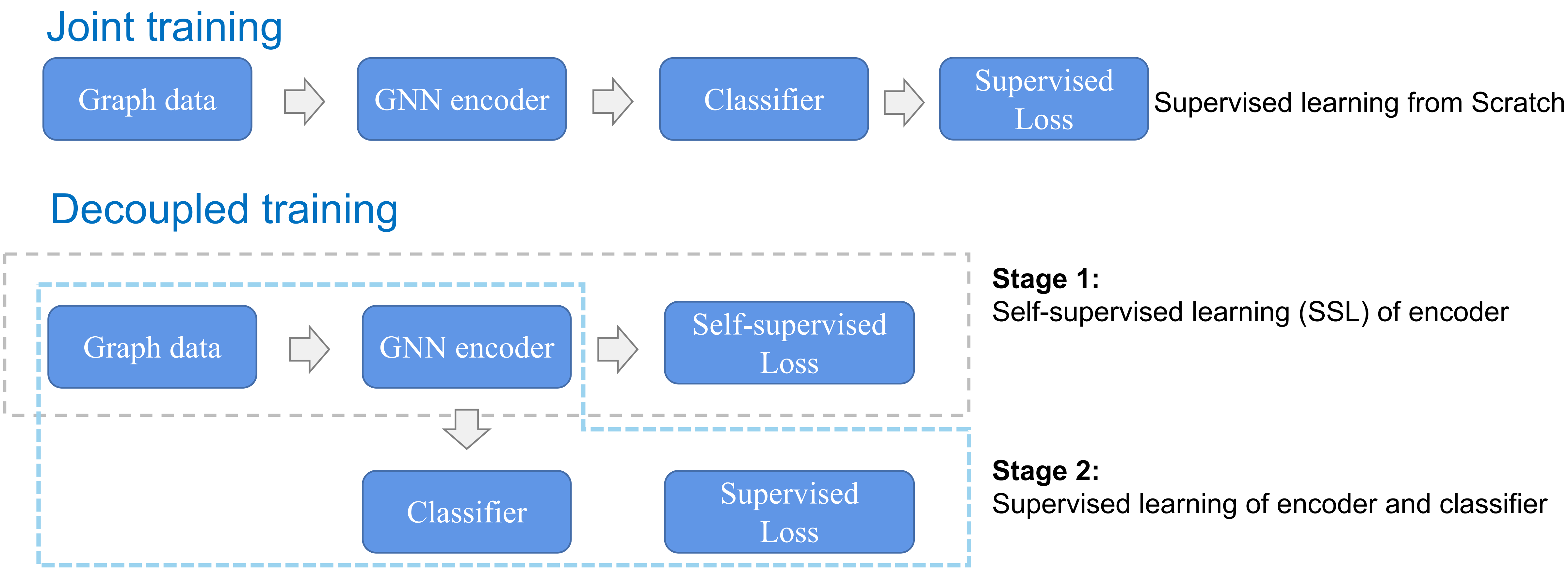
Decoupling Representation Learning and Classification for GNN-based Anomaly Detection

Yanling Wang, Jing Zhang, Shasha Guo,

Hongzhi Yin, Cuiping Li, Hong Chen

Accepted by SIGIR'21

Decoupled Representation Learning



Self-supervised Loss

- **DGI -- Contrast-based loss**

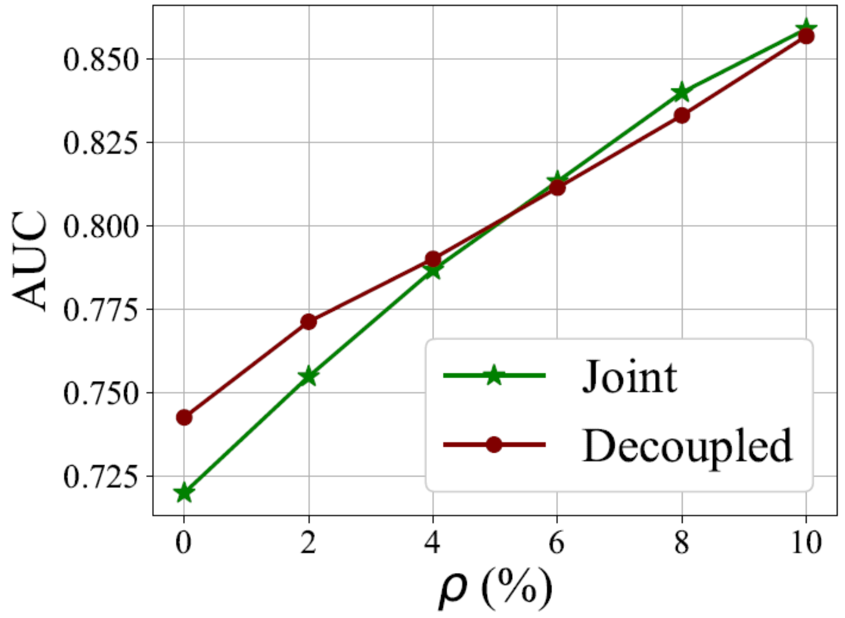
$$\mathcal{L}_{DGI} = -\frac{1}{2n}\sum_{i=1}^{n}\left(\mathbb{E}_{G}\log\mathcal{D}(\boldsymbol{h}_i^{(L)}, \boldsymbol{s}) + \mathbb{E}_{\tilde{G}}\log(1-\mathcal{D}(\tilde{\boldsymbol{h}}_i^{(L)}, \boldsymbol{s}))\right)$$

  - h_i: local node representation
  - s : global graph representation
  - Encodes the global information into node representations to represent the individual behavior patterns as well as the normal pattern occupied by the majority.

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In ICLR.

Why Decoupled Training?

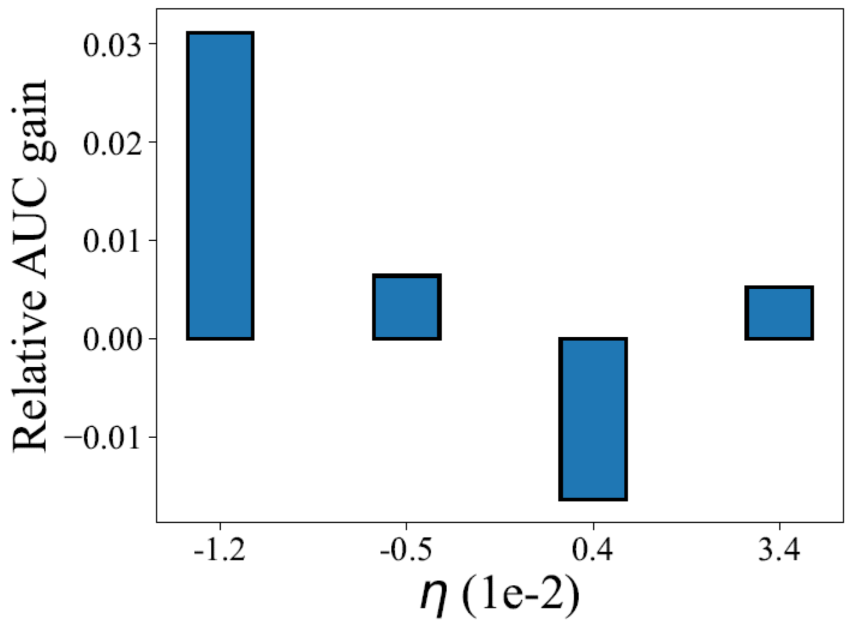
- **Pre-experiments: Joint VS Decoupled over learning difficulty.**
  - Hard instances induced by the inconsistency increase the learning difficulty.
  - Removing ρ hard instances leads to different learning difficulties.
  - A smaller ρ corresponds to a harder case.



Observation:
Decoupled training helps address hard instances.

Is Decoupled Training Stably Better?

- **Pre-experiments: Joint VS Decoupled over inconsistency**
  - Inconsistency is a key factor that impacts the performance of anomaly detection.
  - We use the additive inverse of silhouette coefficient to quantify the inconsistency η.

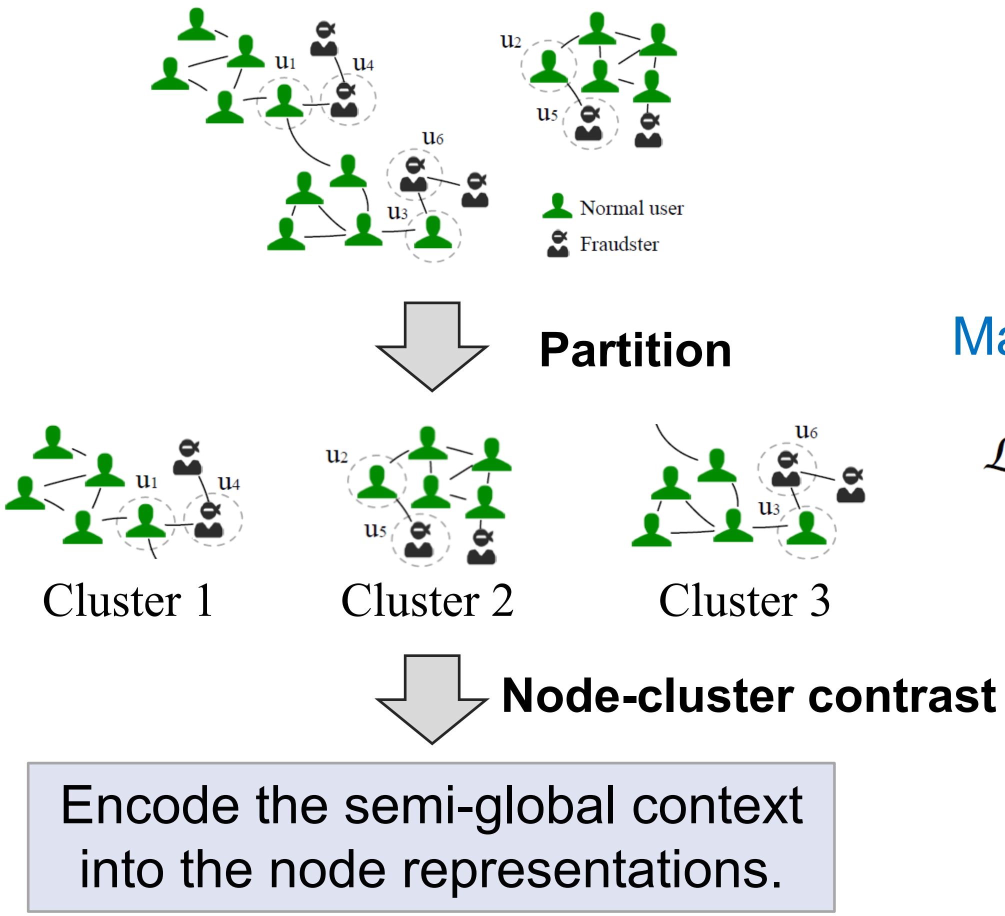


$$
\begin{aligned}
\eta &= -\frac{1}{|V|}\sum_{i=1}^{|V|}\frac{b_i - a_i}{\max\{b_i, a_i\}},\\
a_i &= \frac{1}{|V_i|}\sum_{v_j \in V_i}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, V_i = \{v_j : y_j = y_i\},\\
b_i &= \frac{1}{|\bar{V}_i|}\sum_{v_j \in \bar{V}_i}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \bar{V}_i = \{v_j : y_j \neq y_i\},
\end{aligned}
$$

Observation:
Decoupled training may not always improve, and even brings negative influence when the data gets highly inconsistent.

The Proposed SSL Scheme -- DCI

- **Deep Cluster Infomax (DCI)**



Cluster representation:

$$s_k = \sigma\left(\frac{1}{n_k}\sum_{v_i \in V_k} \boldsymbol{h}_i\right)$$

Maximize the observed local-semi-global affinity scores:

$$\mathcal{L}_{DCI}^{k} = -\frac{1}{2n_k}\sum_{v_i \in V_k}\left(\mathbb{E}_{C_k}\log\mathcal{D}(\boldsymbol{h}_i, \boldsymbol{s}_k) + \mathbb{E}_{\tilde{C}_k}\log(1-\mathcal{D}(\tilde{\boldsymbol{h}}_i, \boldsymbol{s}_k))\right)$$

$$\mathcal{L}_{DCI} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{DCI}^{k}$$

The Proposed SSL Scheme -- DCI

Algorithm 1: Deep Cluster Infomax

Input : Graph $G = (V, A, X)$, Number of clusters K, Number of training epochs t, Number of re-clustering epochs $\bar{t}$.

Output : Optimized GNN encoder g

Initialize clusters $[C_1, C_2, \cdots, C_K] = \text{K-Means}(X)$;

Initialize the parameters θ and ω for the encoder g and the discriminator $\mathcal{D}$;

for *epoch* $\leftarrow 1$ **to** t **do**

  $H = g(G, \theta)$;

  $\mathcal{L}_{DCI} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{DCI}^{k}(H, C_k, \omega)$;

  $\theta, \omega \leftarrow \text{Adam}(\mathcal{L}_{DCI})$;

  if $t \bmod \bar{t} == 0$ **then**

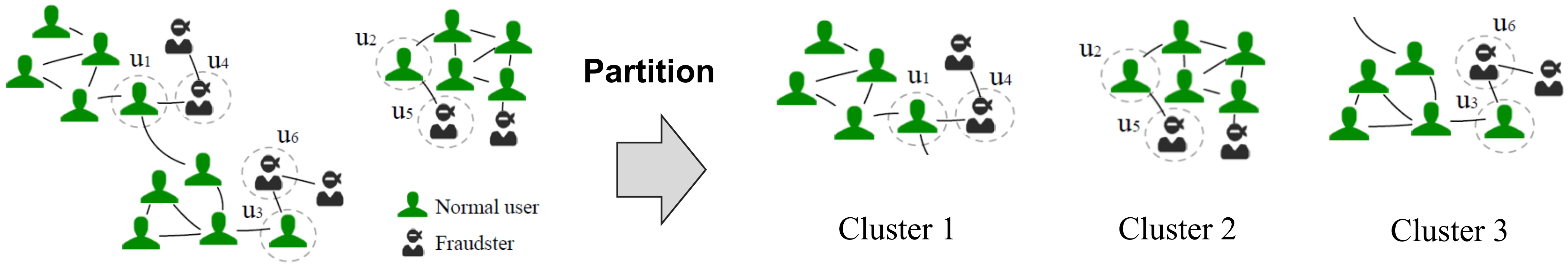
    $[C_1, C_2, \cdots, C_K] = \text{K-Means}(g(G, \theta))$

Return : encoder g

In practice, we re-cluster the nodes based on the node representations after every $\bar{t}$ training epochs.

The Proposed SSL Scheme -- DCI

- **Why can DCI work?**
  - The behavior patterns within the same cluster are often much more concentrated than those in the whole graph.
  - The distance between the users with the opposite labels but close behavior patterns is amplified when the context is restricted into a small cluster



Experiments

- **Datasets**

Table 1: Statistics of the datasets.

Graph	#Users(% normal, abnormal)	#Objects	#Edges
Reddit	10,000 (96.34%, 3.66%)	984	78,516
Wiki	8,227 (97.36%, 2.64%)	1,000	18,257
Alpha	3,286 (61.21%, 38.79%)	3,754	24,186
Amazon	27,197 (91.73%, 8.27%)	5,830	52,156

- **Evaluation**
  - 10-fold/5-fold evaluation.
  - Averaged best AUC score over different folds.

Overall evaluation

Table 2: Overall evaluation on four real-world datasets.

		Reddit	Wiki	Alpha	Amazon
Joint	CARE-GNN	0.700	0.702	0.802	0.729
	GAT	0.738	0.681	0.848	0.696
	GeniePath	0.720	0.689	0.849	0.738
	GIN	0.720	0.727	0.884	0.761
Decoupled	GAE	0.730	0.714	0.884	0.806
	RW	0.728	0.740	**0.908**	0.782
	GCC	0.669	0.695	0.865	0.733
	DGI	0.743	0.737	0.884	0.771
	DCI (ours)	**0.746**	**0.762**	0.907	**0.810**
Inconsistency η (1e-2)		-0.676	0.841	-	-

Note: All the decoupled models use GIN's encoder as the backbone.

(1) Decoupled training contributes to the anomaly detection.

(2) DCI is an effective SSL scheme for decoupled training.

(3) Decoupled training with DCI shows promising performance on the more inconsistent dataset Wiki.

Comparison with the Multi-task Learning

Table 4: Evaluation of the multi-task learning.

		Reddit	Wiki	Alpha	Amazon
Joint	GIN	0.720	0.727	0.884	0.761
Multi-task	GAE	0.726	0.705	0.904	0.766
	DGI	0.647	0.664	0.891	0.806
	DCI	0.675	0.670	0.893	0.803

Note: All the multi-task models use GIN's encoder as the backbone.

Table 2: Overall evaluation on four real-world datasets.

		Reddit	Wiki	Alpha	Amazon
Joint	CARE-GNN	0.700	0.702	0.802	0.729
	GAT	0.738	0.681	0.848	0.696
	GeniePath	0.720	0.689	0.849	0.738
	GIN	0.720	0.727	0.884	0.761
Decoupled	GAE	0.730	0.714	0.884	0.806
	RW	0.728	0.740	**0.908**	0.782
	GCC	0.669	0.695	0.865	0.733
	DGI	0.743	0.737	0.884	0.771
	DCI (ours)	**0.746**	**0.762**	0.907	**0.810**
Inconsistency η (1e-2)		-0.676	0.841	-	-

Note: All the decoupled models use GIN's encoder as the backbone.

(1) Multi-task learning can outperform the joint training, but does not always bring improvements.

(2) Decoupled training shows advantages over the multi-task learning.

(3) Compared with GAE, DGI and DCI allow to learn more implicit and expressive structural patterns.

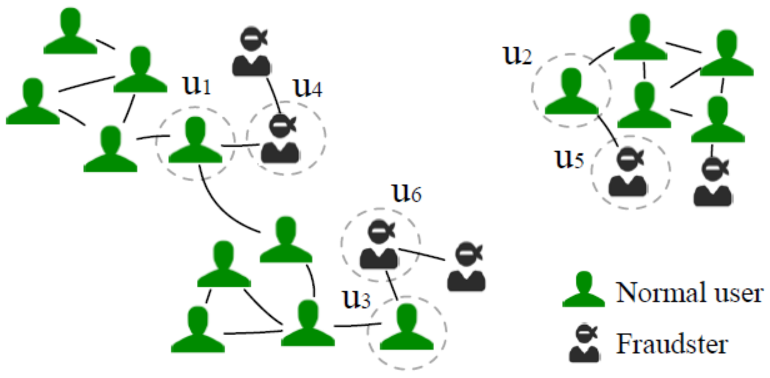
So using $\mathcal{L}_{DGI}$ or $\mathcal{L}_{DCI}$ could amplify the inconsistency, making multi-task learning vulnerable.

Contributions

- Our study reveals an intriguing phenomenon -- inconsistency between the behavior patterns and the label semantics highly impacts the performance of graph embedding.
- We suggest that decoupled training equipped with a proper SSL objective can be an alternative way for effective anomaly detection.
- We propose an effective SSL scheme called DCI for anomaly detection.
- The findings and proposed model here is not stricted to anomaly detection.

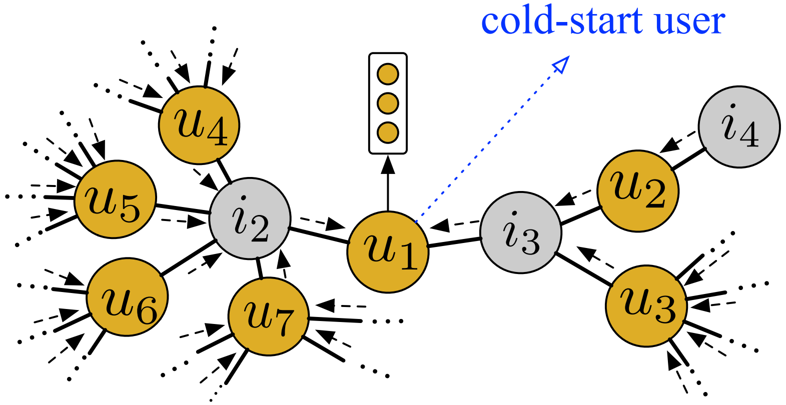
End-to-End Training Suffers from Limitations

- Data Inconsistency
  - The node patterns and the label semantics disagree with each other.
  - Dilemma: to learn the intrinsic graph properties or to capture the label semantics?



Anomaly Detection

- Sparse Interactions
  - The nodes have sparse interactions with others
  - Prevent learning high-quality embeddings.



Recommendation

Pre-Training Graph Neural Networks for Cold-Start Users and Items Representation

Bowen Hao, Jing Zhang,

Hongzhi Yin, Cuiping Li, Hong Chen

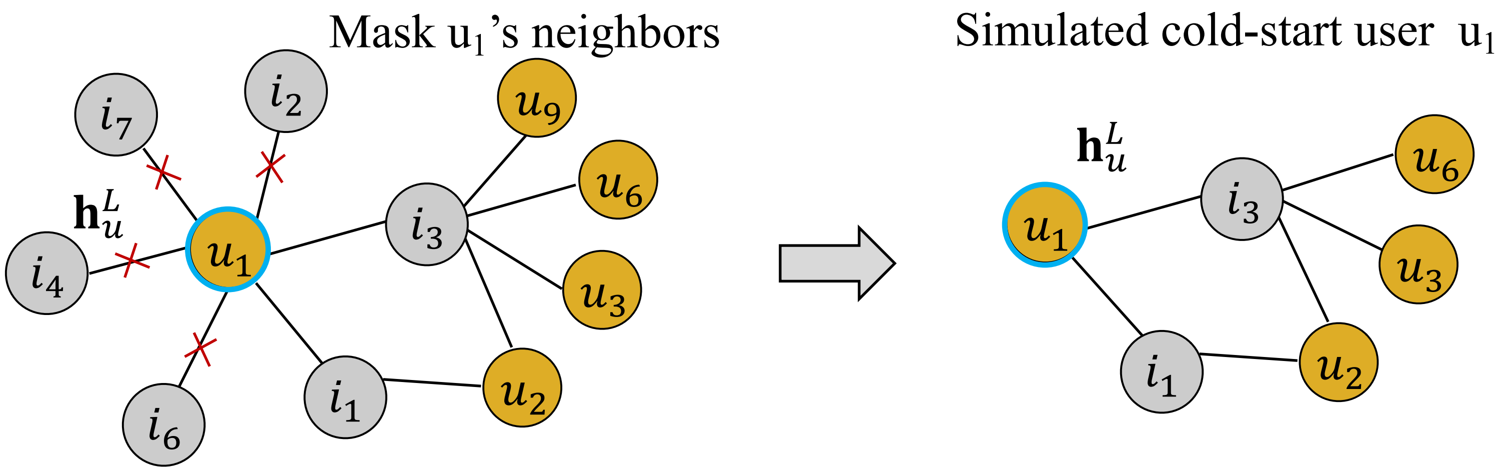
Accepted by WSDM'21

Self-Supervised Loss

- **Cold-start User Representation Reconstruction -- Generation-based loss**

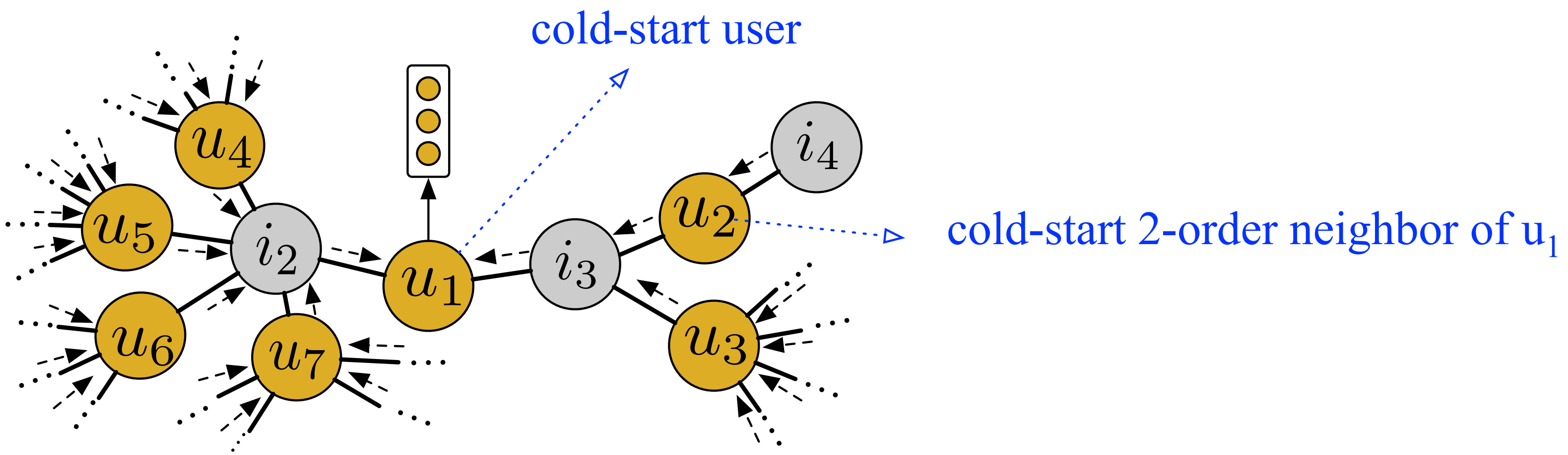
$$\Theta_f^* = \arg\max_{\Theta_f} \sum_u \cos(\mathbf{h}_u^L, \mathbf{h}_u),$$

- Simulate cold-start users by normal users (by masking neighbors).
- Learn ground truth representation h_u from u's abundant interactions.
- Reconstruct the ground truth representation from the simulated cold-start users.

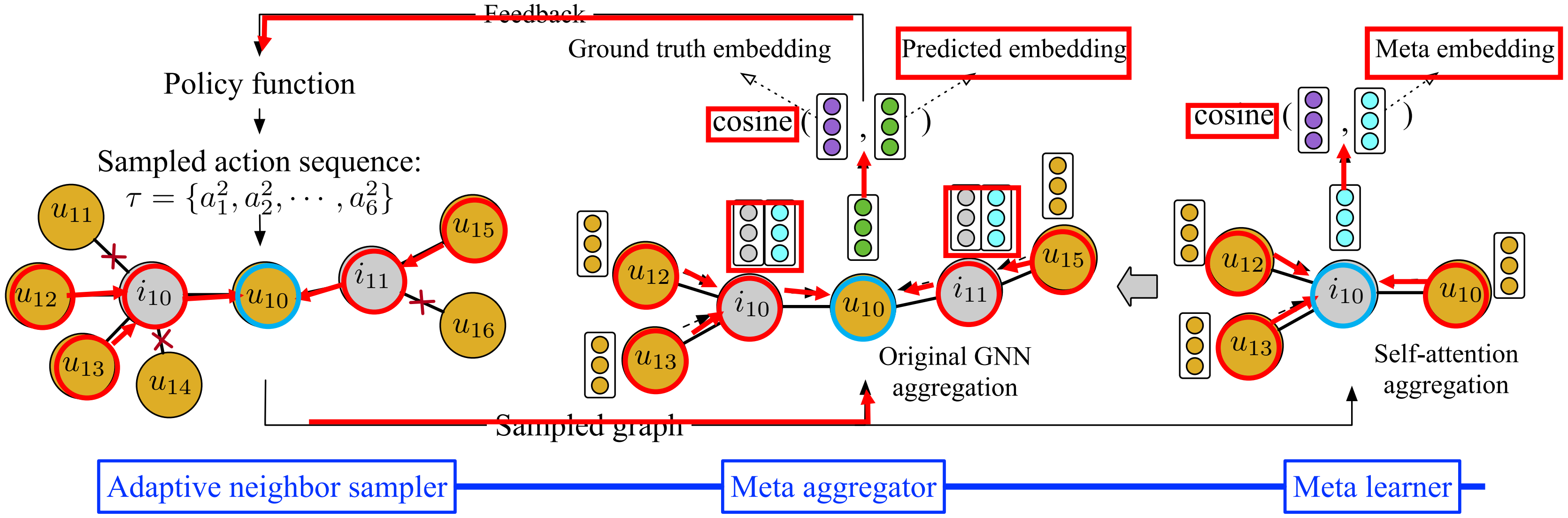


Remaining Issue of GNN

- The cold-start neighbors are not explicitly dealt with during graph convolution.
- The random or importance sampling strategy fail to sample high-order relevant cold-start neighbors due to their sparse interactions.



Enhanced Pre-Training GNN Model



Pre-train the GNN model for reconstructing embeddings

Action: {Remove, keep} **Policy function:** two-layer Neural Network

$$\mathbf{H}_t^l = \mathrm{ReLU}(\mathbf{W}_1^l \mathbf{s}_t^l + \mathbf{b}^l)$$

$$P(a_t^l|\mathbf{s}_t^l, \Theta^l) = a_t^l \sigma(\mathbf{W}_2^l \mathbf{H}_t^l) + (1 - a_t^l)(1 - \sigma(\mathbf{W}_2^l \mathbf{H}_t^l))$$

State s_t^l: Similarity between a neighbor and the target user

Reward:

$$R(a_t^l, \mathbf{s}_t^l) = \begin{cases} \cos(\hat{\mathbf{h}}_u^L, \mathbf{h}_u) - \cos(\mathbf{h}_u^L, \mathbf{h}_u) & \text{if } t = |\mathcal{N}^{l'}(u)| \wedge l = l'; \\ 0 & \text{otherwise,} \end{cases}$$

Enhanced Pre-Training GNN Model Process

Algorithm 2: The Overall Training Process.

Pre-train the meta learner with parameter Θ_g;
Pre-train the meta aggregator with parameter Θ_f when fixing Θ_g;
Pre-train the neighbor sampler with parameter Θ_s by Algorithm 1 when fixing Θ_g and Θ_f;
Jointly train the three modules together with parameters Θ_g, Θ_f and Θ_s by running Algorithm 1;

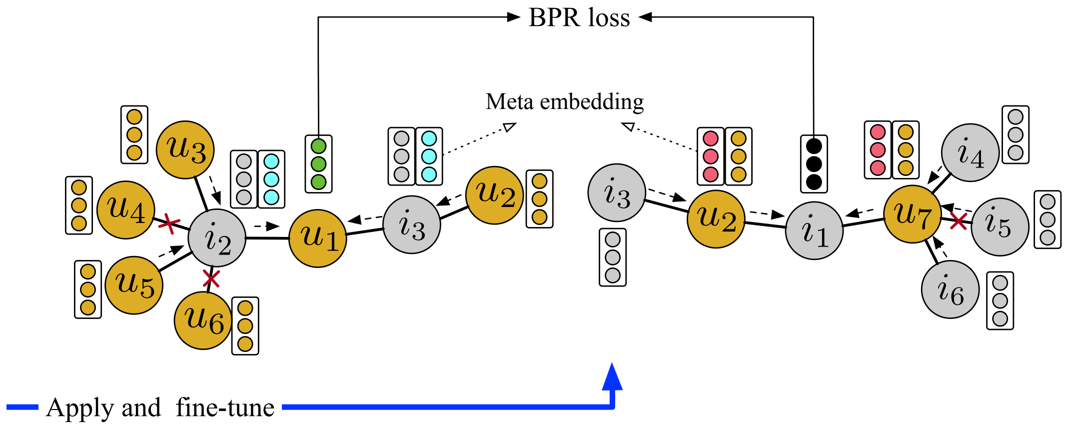
Algorithm 1: The Joint Training Process.

Input: $Train_T = \{(u_k, i_k)\}$, the ground truth embeddings $\{(\mathbf{h}_u, \mathbf{h}_i)\}$, a pre-trained meta learner with Θ_g^0, meta aggregator with Θ_f^0 and Θ_g^0 and neighbor sampler with Θ_s^0.

```
Initialize Θs = Θs^0, Θf = Θf^0, Θg = Θg^0 ;
for epoch from 1 to E do
    foreach u_k or i_k in Train_T do
        for l in {2, 3, ··· , L} do
            Sample a sequence of actions
              τ^l = {a_1^l, ··· , a_t^l, ··· , a_{|N^l(u)|}^l} by Eq. (7);
            if ∀ a_t^l = 0 or l = L then
                Compute R(a_{|N^l(u)|}^l, s_{|N^l(u)|}^l) by Eq. (8);
                Compute gradients by Eq. (9);
                Break;
    Update Θs;
    if Jointly Training then
        Update Θg and Θf ;
```

Adaptive Neighbor Sampling

Model Fine-tuning



Fine-tune the GNN model for recommendation

- Sample u's neighbors $\{\mathcal{N}^1(u), \widehat{\mathcal{N}}^2(u), \ldots, \widehat{\mathcal{N}}^L(u)\}$ based on his original L-order neighbors $\{\mathcal{N}^1(u), \ldots, \mathcal{N}^L(u)\}$
- Obtain aggregated user/item embedding $\mathbf{h}_u^L$, $\mathbf{h}_i^L$
- Calculate the relevant score $y(u, i) = \sigma(\mathbf{W} \cdot \mathbf{h}_u^L)^T \sigma(\mathbf{W} \cdot \mathbf{h}_i^L)$
- Use BPR loss to optimize the model parameters

Experimental Settings

- Intrinsic evaluation
  - Predict user/item embeddings
  - Spearman Correlation
  - D_T
- Extrinsic evaluation
  - Recommendation
  - Recall@K, NDCG@K
  - D_N
- Baselines
  - NCF
  - GraphSAGE, GAT, FastGCN
  - FBNE, LightGCN

Table 1: Statistics of the Datasets.

Dataset	#Users	#Items	#Interactions	#Sparse Ratio
MovieLens-1M	6,040	3,706	1,000,209	4.47%
MOOCs	82,535	1,302	458,453	0.42%
Last.fm	992	1,084,866	19,150,868	1.78%

Table 1: Details of splitting the Datasets.

Dataset	D_T(user)	D_N(user)	D_T(item)	D_N(item)
MovieLens-1M	$\geq$60	<60	$\geq$60	<60
MOOCs	$\geq$20	<20	$\geq$20	<20
Last.fm	-	-	$\geq$15	<15

Predict User/Item Embeddings

Table 2: Overall performance of user/item embedding inference (Spearman correlation). The layer depth L is 3.

Methods	Ml-1M (user)		MOOCs (user)		Last.fm (user)		Ml-1M (item)		MOOCs (item)		Last.fm (item)	
	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot
NCF	-0.017	0.063	-0.098	-0.062	0.042	0.117	-0.118	-0.017	-0.036	0.027	-0.036	-0.018
GraphSAGE	0.035	0.105	0.085	0.128	0.104	0.134	0.113	0.156	0.116	0.182	0.112	0.198
Basic-GraphSAGE	0.076	0.198	0.103	0.152	0.132	0.184	0.145	0.172	0.172	0.196	0.166	0.208
Meta-GraphSAGE	0.258	0.271	0.298	0.320	0.186	0.209	0.434	0.448	0.288	0.258	0.312	0.333
NSampler-GraphSAGE	0.266	0.284	0.294	0.336	0.196	0.212	0.448	0.460	0.286	0.306	0.326	0.336
GraphSAGE*	**0.368**	**0.375**	**0.302**	**0.338**	**0.326**	**0.384**	**0.470**	**0.491**	**0.316**	**0.336**	**0.336**	**0.353**
GAT	0.020	0.049	0.092	0.138	0.092	0.125	0.116	0.126	0.108	0.118	0.106	0.114
Basic-GAT	0.046	0.158	0.104	0.168	0.158	0.180	0.134	0.168	0.112	0.126	0.209	0.243
Meta-GAT	0.224	0.282	0.284	0.288	0.206	0.212	0.438	0.462	0.294	0.308	0.314	0.340
NSampler-GAT	0.296	0.314	0.339	0.354	0.198	0.206	0.464	0.472	0.394	0.396	0.338	0.358
GAT*	**0.365**	**0.379**	**0.306**	**0.366**	**0.309**	**0.394**	**0.496**	**0.536**	**0.362**	**0.384**	**0.346**	**0.364**
FastGCN	0.009	0.012	0.063	0.095	0.082	0.114	0.002	0.036	0.007	0.018	0.007	0.013
Basic-FastGCN	0.082	0.146	0.083	0.146	0.104	0.149	0.088	0.113	0.099	0.121	0.159	0.182
Meta-FastGCN	0.181	0.192	0.282	0.280	0.224	0.274	0.216	0.266	0.248	0.278	0.230	0.258
NSampler-FastGCN	0.188	0.194	0.281	0.286	0.226	0.277	0.268	0.288	0.267	0.296	0.246	0.253
FastGCN*	**0.198**	**0.212**	**0.288**	**0.291**	**0.266**	**0.282**	**0.282**	**0.298**	**0.296**	**0.302**	**0.268**	**0.278**
FBNE	0.034	0.102	0.053	0.065	0.142	0.164	0.168	0.190	0.137	0.168	0.127	0.133
Basic-FBNE	0.162	0.190	0.162	0.185	0.135	0.180	0.176	0.209	0.157	0.180	0.167	0.173
Meta-FBNE	0.186	0.204	0.269	0.284	0.175	0.192	0.426	0.449	0.236	0.272	0.178	0.182
NSampler-FBNE	0.208	0.216	0.259	0.283	0.203	0.207	0.422	0.439	0.226	0.273	0.164	0.183
FBNE*	**0.242**	**0.265**	**0.306**	**0.321**	**0.206**	**0.219**	**0.481**	**0.490**	**0.301**	**0.382**	**0.182**	**0.199**
LightGCN	0.093	0.108	0.060	0.068	0.162	0.184	0.201	0.262	0.181	0.232	0.213	0.245
Basic-LightGCN	0.178	0.192	0.212	0.226	0.182	0.192	0.318	0.336	0.234	0.260	0.252	0.290
Meta-LightGCN	0.226	0.241	0.272	0.285	0.206	0.221	0.336	0.346	0.314	0.331	0.372	0.392
NSampler-LightGCN	0.238	0.256	0.286	0.294	0.204	0.212	0.348	0.384	0.296	0.314	0.356	0.401
LightGCN*	**0.270**	**0.286**	**0.292**	**0.309**	**0.229**	**0.234**	**0.382**	**0.408**	**0.334**	**0.353**	**0.386**	**0.403**

- vs. NCF, GNNs incorporate high-order neighbors.
- vs. GNNs, basic pre-training GNNs explicitly deal with the cold-start users/items.
- vs. basic pre-training GNNs, incorporating the meta aggregator explicitly deal with high-order cold-start users/items, and the neighbor sampler can sample high-order relevant cold-start neighbors.
- When K decreases from 8 to 3, the performance gain is more significant.

Contributions

- A pre-training GNN model via reconstructing cold-start user/item embeddings to explicitly improve the embedding quality of users/items.

- Incorporate a meta learner to enhance cold-start neighbors' embeddings, and a neighbor sampler to sample relevant high-order neighbors.

- Experiments on three real-world datasets show the superiority of our pre-training model compared with state-of-the-art GNN models.

Conclusions

- Exploited potential self-supervised GNN strategies to solve
  - Data consistency
    - By build a semi-global SSL objective.
  - Sparse Interactions
    - By build a reconstruction SSL objective under meta-learning setting.

- To Discuss
  - Is there a unified guidance for designing the SSL objectives of different downstream tasks?
  - How to enable the transferability of the pre-trained GNN encoder?

Thank You

Q&A