GEMS LAB

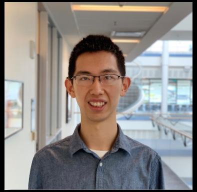
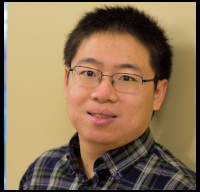
COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

DEEP GRAPH LIBRARY

SpotTarget: Rethinking the Effect of Target Edges for Link Prediction in Graph Neural Networks

Jing Zhu*, Yuhang Zhou*, Vassilis Ioannidis, Shengyi Qian, Wei Ai, Xiang Song, Danai Koutra

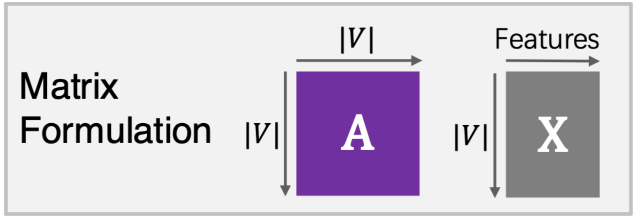


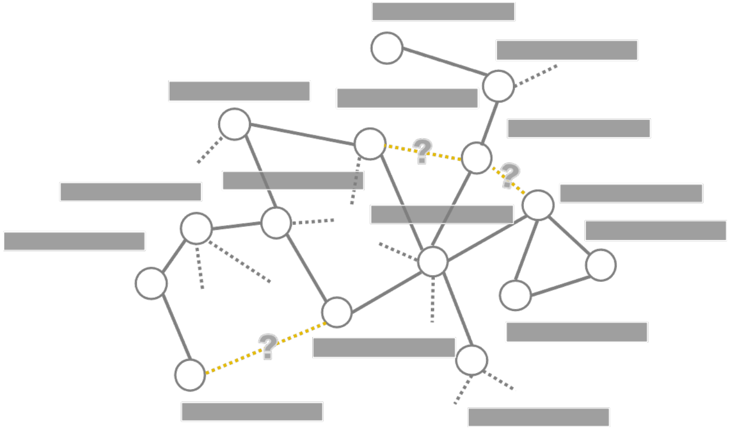




MLG – Aug.7 2023

Link Prediction (LP)

- Given a graph $\mathcal{G} = (V, E)$ with known edges E represented in adjacency matrix $\mathbf{A}$; feature vector $\mathbf{x}$ for each node;
- Find other potential edges in the graph

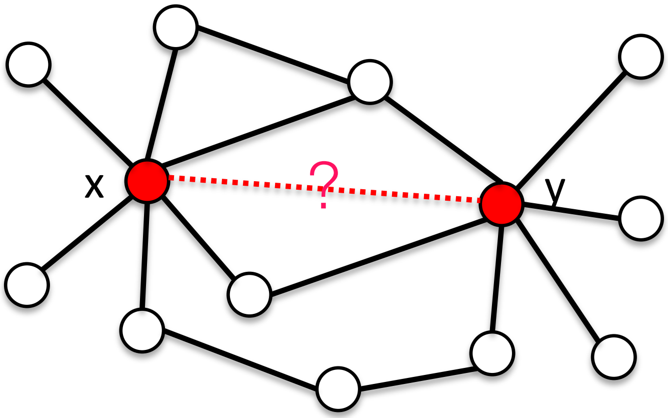




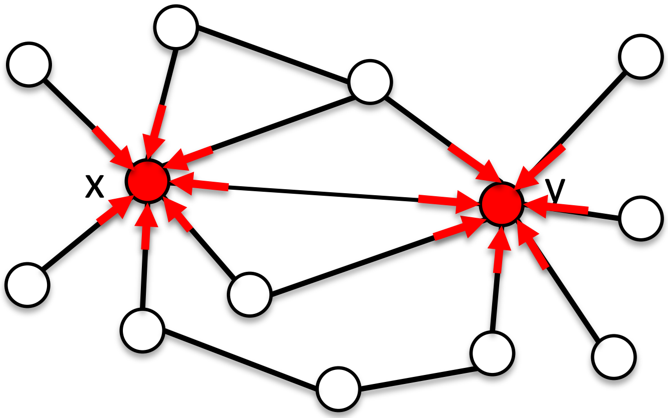
Applications:

- **Learn embeddings for a variety of downstream tasks: query response, reducing spam, universal embeddings, ...**
- **Specific link prediction applications: graph completion, …**

Dual Roles For Edges in LP



As prediction target



Message Passing

Common practice: include target links in the message passing graph at training and/or testing time

Figure credit: Muhan Zhang (wlnm)

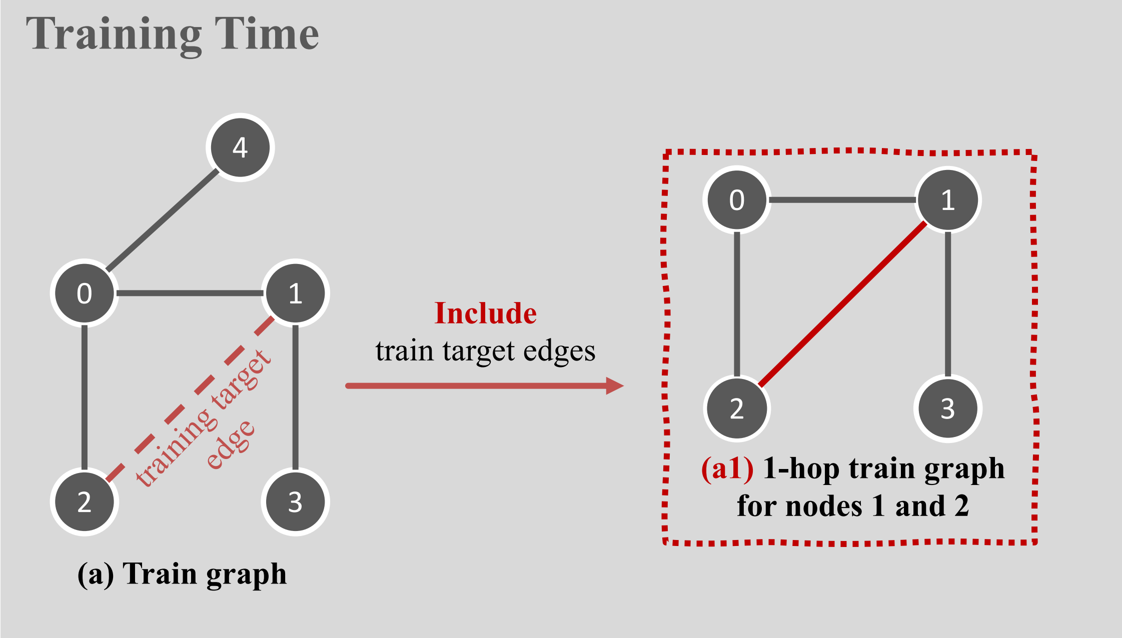
Previous Works

- Most discussion about target edge inclusion falls into subgraph-based methods at training time
  - SEAL: noticed the inclusion of target links at training and proposed negative injection
  - FakeEdge: discussed the distribution shift issue and resolved it via always adding or removing the target links, or combining the strategies
- Here, we aim to show simply excluding all target links does not fully solve the problem for both GAE and subgraph-based method. We further extend the target edge inclusion discussion to test time.

Contributions

- Systematic Analysis of the Target Link Inclusion Practices: We propose first thorough theoretical and empirical analysis on the effect of including target edges as message-passing edges at training and test time.
- Efficient Unified Framework: We propose SpotTarget, which automatically tackles these issues. We integrate this as a plug-and-play framework into DGL.
- Extensive Experiments: We show that SpotTarget makes GNN models up to 15× more accurate on sparse graphs, and significantly improves their performance for low-degree nodes on dense graphs.

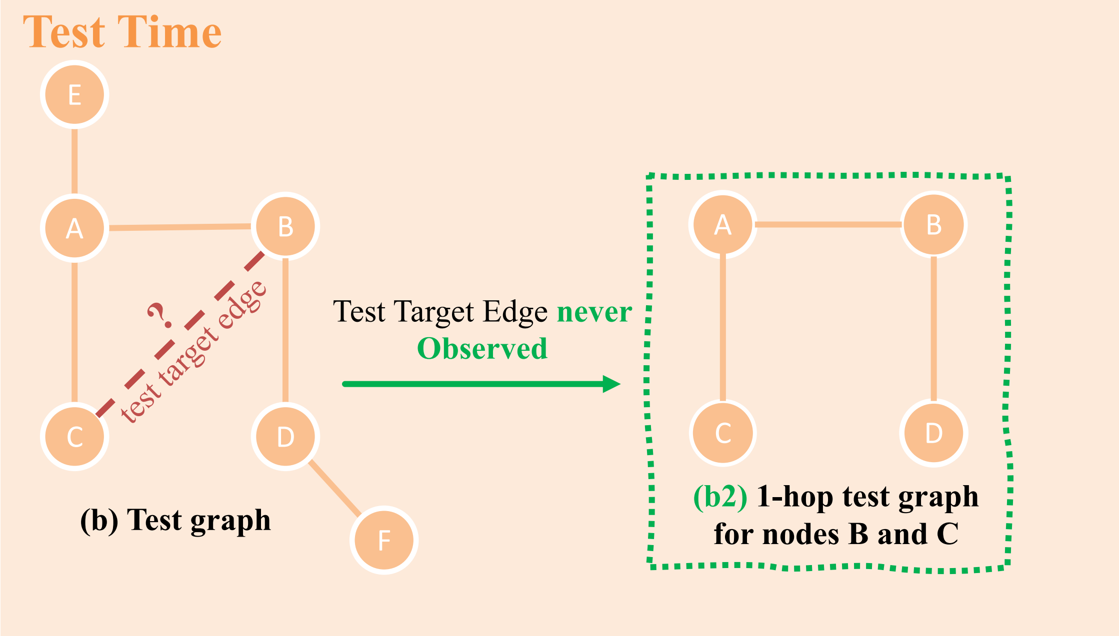
Training Pitfalls



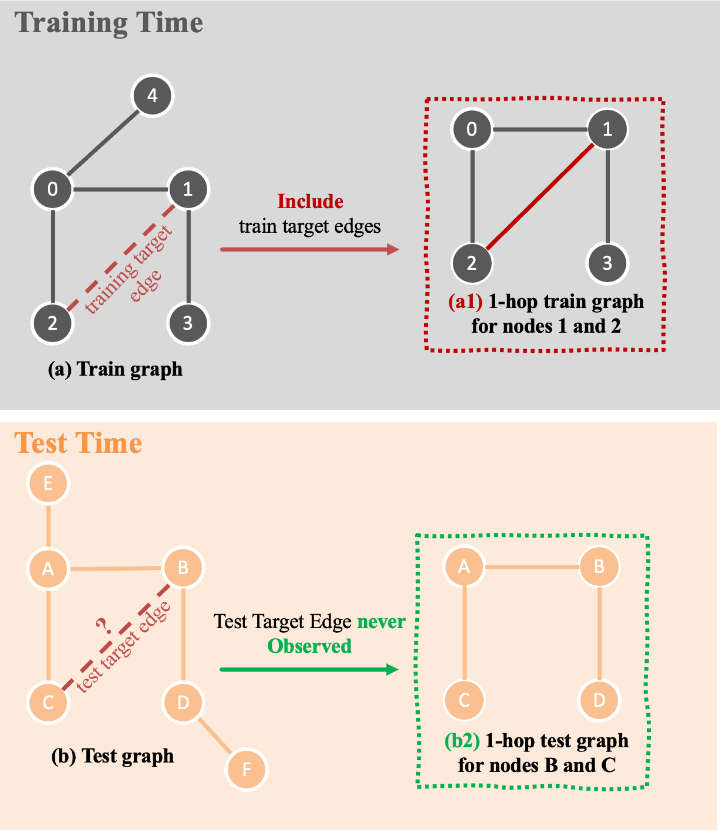
Training I1:
Overfitting (a1)

Train prediction targets can be seen in the graph

Training Pitfalls



Training Pitfalls



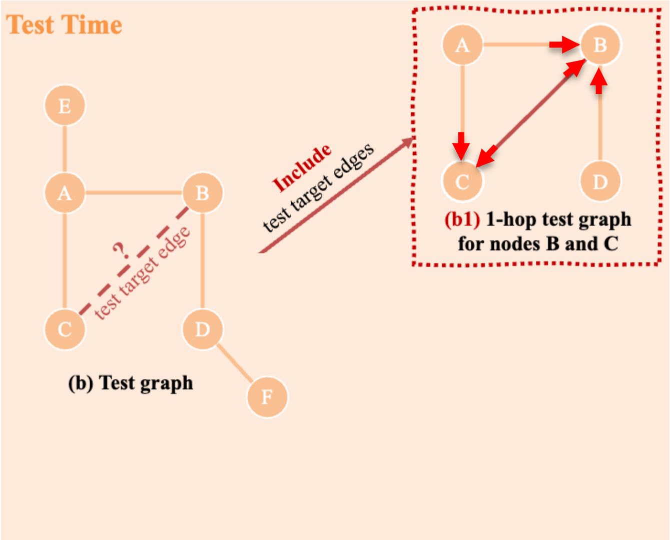
Training I2: Distribution shift (a1, b2):

Discrepancy between the graphs used during training and test

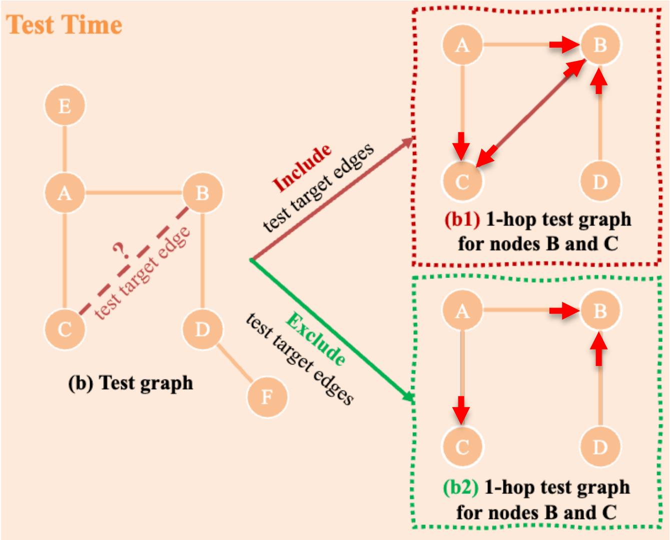
Poor GNN generalizability!

Jing Zhu et. al

Test Pitfalls



Test Pitfalls



Test Pitfalls I3 Data Leakage (b1):

If test target edges exist in the MP graph, it results in higher likelihood of predicting target edges existence

Overestimation of the model's predictive performance!

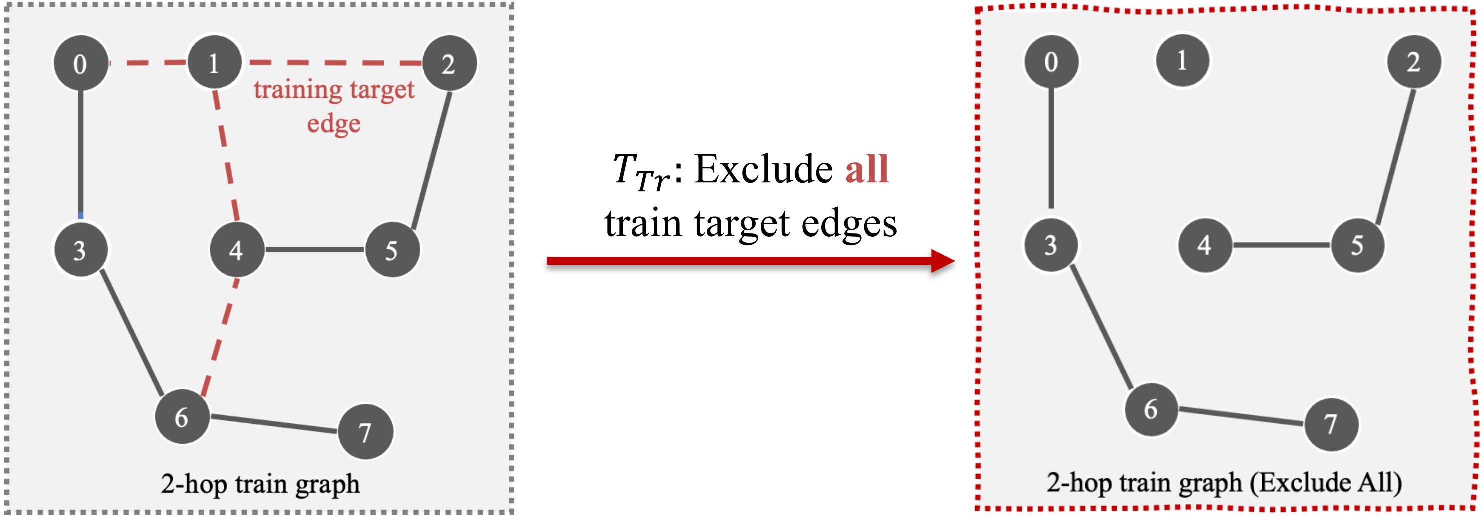
Goal

Goal: Given a graph G, a link prediction task, and a base GNN model in a mini-batch training setting, design a framework that proposes solutions to best avoid the training and test pitfalls I1, I2 and I3.

One straightforward solution is to exclude all target edges.

At training time, naïve solution does not work well!

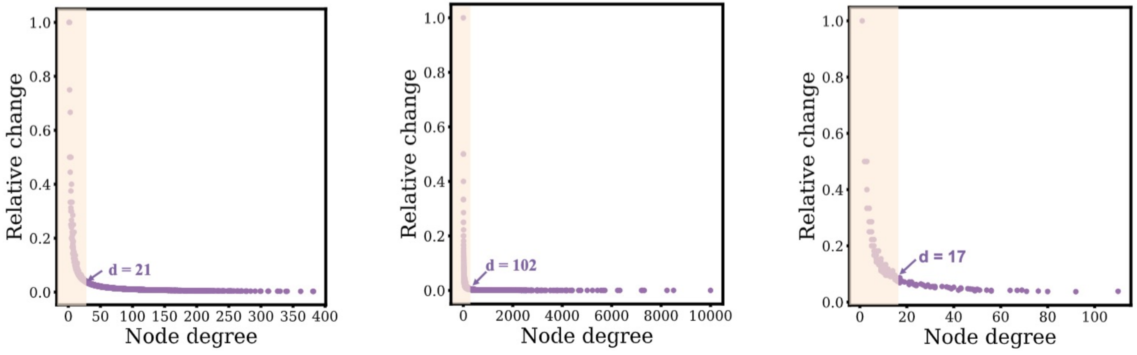
Naive Solution Issues



- Graph Structure Corruption for the MP graph when ExcludeAll (isolated components, isolated nodes).
- Batch size=1, too small MP graph, inefficiency and instability for GNN training.

Real Question

How can we achieve the best trade-off between avoiding issues (I1)-(I2) and preserving the graph structure in mini-batch training as much as possible?



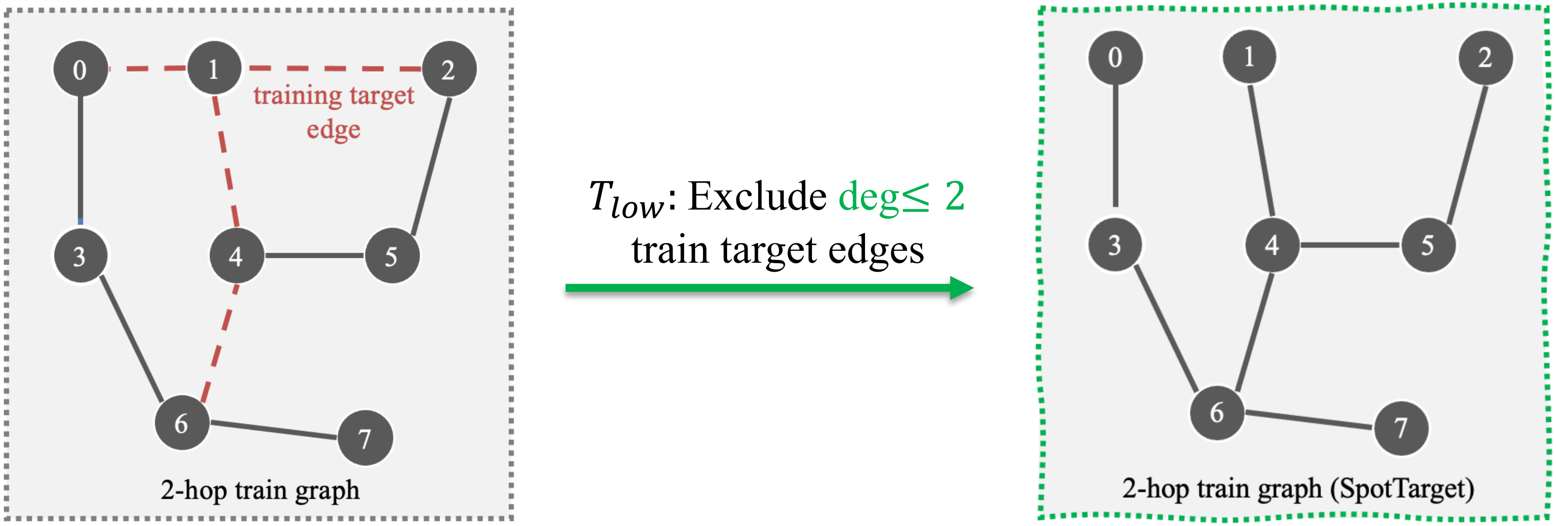
(a) Ogbl-Collab (b) Ogbl-Citation2 (c) USAir (d) E-commerce

Lower-degree nodes have higher relative degree change before and after excluding all train target links in each mini-batch.

Jing Zhu et. al

Training Time Right Practice

- For high degree nodes, issues from one neighboring nodes (I1,I2) are diluted and tend not to affect much.
- Only exclude the training target edges (T_{low}) incident to at least one low-degree node.



Jing Zhu et. al

Theoretical Analysis

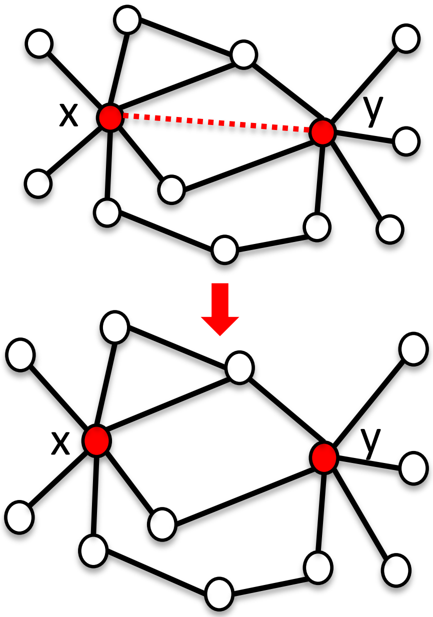
- The change in influence that a random node v_k has on a high-degree node v_h and a low-degree node v_l before and after excluding an edge incident to v_h and v_l, is higher on v_l.

Let v_h and v_l be two nodes in a graph with degrees $d_h > d_l$, and node v_k be an arbitrary node in the graph. Assume that ReLU is the activation function, the Λ-layer GNN is untrained, and all random walk paths have a return probability of 0. We denote the effect of node v_k on node v_h after Λ-th layer GNN as $x_h^\Lambda x_k$ where x_h, x_k are $n-$dimensional vectors indicating the embeddings for nodes v_h, v_k, respectively. Further we denote that effect of node v_k on node v_h after removing an incident edge to node v_h as $\tilde{x_h}^\Lambda x_k$. We define the change in effect of v_k on v_h before and after removing an incident edge to v_h as distance function $D(k,h) = 1 - \mathbb{E}(\tilde{x}_{h,s}^\Lambda x_{k,t} / x_{h,s}^\Lambda x_{k,t})$ for any entry $1 \leq s,t \leq n$ of x_h and x_k. Similarly, we define the change in effect of node v_k on v_l as $D(k,l) = 1 - \mathbb{E}(\tilde{x}_{l,s}^\Lambda x_{k,t} / x_{l,s}^\Lambda x_{k,t})$ for any entry $1 \leq s,t \leq n$ of x_l and x_k. Then, $D(k,h) < D(k,l)$.

Jing Zhu et. al

Test Time Right Practice

- Exclude all test target links to prevent the data leakage.
- Implementation: A module that automatically checks for the presence of test target edges in the inference graph and removes them if necessary.



Algorithm 1 SpotTarget: Leakage Check(G)

Input: An input graph $\mathbf{G}$, edge splits $\mathbf{S}$, an argument $\mathbf{K}$ if valid edges are used as inference inputs, $\mathbf{K} = \{T, F\}$
Output: The desired inference graph $\mathbf{G}_{\text{infer}}$
// STEP 1. Check if the input graph contains validation and test edges
C_{valid} = Check Existence($\mathbf{G}, \mathbf{S}_{\text{valid}}$)
C_{test} = Check Existence($\mathbf{G}, \mathbf{S}_{\text{test}}$)
// STEP 2. Delete test and validation edges according to user requirement
if C_{test} is True **then**
    $\mathbf{G}_{\text{infer}}$ = RemoveEdge($\mathbf{G}, \mathbf{S}_{\text{test}}$)
else
    $\mathbf{G}_{\text{infer}} = \mathbf{G}$
// If Validation edges exist in the inference graph and it is not desired
if C_{valid} is True and $\mathbf{K}$ is False **then**
    $\mathbf{G}_{\text{infer}}$ = RemoveEdge($\mathbf{G}_{\text{infer}}, \mathbf{S}_{\text{valid}}$)
return $\mathbf{G}_{\text{infer}}$

Experiments

- Q1: How well does SpotTarget address issues (I1) and (I2) on commonly-used graph benchmarks, which are dense?
- Q2: How well does SpotTarget perform on sparse graphs with very skewed degree distributions?
- Q3: How well does SpotTarget address issues (I1)-(I2) for edges incident to low-degree nodes on popular benchmarks?
- Q4: At test time, how much is the performance of GNN models overestimated due to implicit data leakage (I3)?

Dataset	# Nodes	# Edges	Node deg.	Attr. dim.
ogbl-collab [12]	235,868	2,358,104	8.20	128
ogbl-citation2 [12]	2,927,963	30,387,995	20.73	128
USAir [26]	332	3,402	10.25	332
E-commerce [25]	346,439	238,818	1.38	768

Training Pitfalls: Results on Dense Data

Model	ExcludeNone(Tr)	ExcludeAll	SpotTarget
	Ogbl-Collab (H@50 ↑)		
SAGE	48.57 ± 0.74	45.82 ± 0.41	49.00 ± 0.65
MB-GCN	43.03 ± 0.50	37.75 ± 1.42	39.58 ± 1.06
GATv2	45.61 ± 0.85	45.71 ± 0.87	45.46 ± 0.19
SEAL	61.27 ± 0.28	64.11 ± 0.30	64.57 ± 0.30
	Ogbl-Citation2 (MRR ↑)		
SAGE	82.06 ± 0.06	81.47 ± 0.17	82.18 ± 0.18
MB-GCN	79.70 ± 0.25	79.06 ± 0.30	79.88 ± 0.14
GATv2	OOM	OOM	OOM
SEAL	86.75 ± 0.20	86.74 ± 0.23	86.93 ± 0.55
	USAir (AUC ↑)		
SAGE	95.97 ± 0.17	95.71 ± 0.12	96.19 ± 0.53
MB-GCN	94.00 ± 0.14	94.09 ± 0.11	94.28 ± 0.15
GATv2	95.05 ± 0.66	95.66 ± 0.24	95.87 ± 0.46
SEAL	95.36 ± 0.24	95.94 ± 0.04	96.39 ± 0.09
Rank ↓	2.27	2.45	1.27

- Across all datasets and models, SpotTarget achieves the best results compared with ExcludeNone(Tr) and ExcludeAll.
- In many cases (6/11), ExcludeAll leads to performance degradation because of corrupting the structure of mini-batch graphs.

FakeEdge

Jing Zhu et. al

Training Pitfalls: Results on Sparse Data

Metrics	SAGE		MB-GCN		GATv2	
	ExcludeNone(Tr)	SpotTarget	ExcludeNone(Tr)	SpotTarget	ExcludeNone(Tr)	SpotTarget
MRR ↑	4.40 ± 0.31	65.85 ± 0.31	17.07 ± 7.38	69.67 ± 0.52	5.98 ± 0.56	69.44 ± 0.55
H@10 ↑	6.55 ± 0.37	89.67 ± 0.19	28.35 ± 7.47	89.79 ± 0.25	9.64 ± 1.10	90.52 ± 0.26
H@1 ↑	3.04 ± 0.31	52.84 ± 0.46	10.83 ± 5.21	57.63 ± 0.57	3.94 ± 0.81	57.11 ± 1.03

- SpotTarget achieves 14.9× better performance compared to ExcludeNone across models.
- This verifies empirically that low-degree nodes suffer more from issues I1 and I2, and excluding T_{low} works well especially for datasets with many low-degree nodes.

Jing Zhu et. al

Training Pitfalls: Results on T_{low}

	Exclusion	$max(d_i, d_j) < 10$	$max(d_i, d_j) < 5$	$min(d_i, d_j) < 10$	$min(d_i, d_j) < 5$	$min(d_i, d_j) = 2$	$min(d_i, d_j) = 1$
	ExcludeNone(Tr)	73.11 ± 0.25	62.15 ± 0.84	78.78 ± 0.12	69.54 ± 0.37	47.02 ± 0.56	27.54 ± 0.88
MRR ↑	**ExcludeAll**	77.45 ± 0.41	75.39 ± 1.42	79.17 ± 0.12	73.86 ± 0.33	60.05 ± 1.11	48.60 ± 1.11
	SpotTarget	78.08 ± 0.06	76.23 ± 0.56	79.30 ± 0.18	73.87 ± 0.18	61.48 ± 0.51	51.47 ± 2.51

- Comparing with ExcludeNone(Tr) and ExcludeAll, SpotTarget achieves better performance on various types of edges that are incident to low-degree nodes.

Test Pitfalls: Data Leakage Quantification

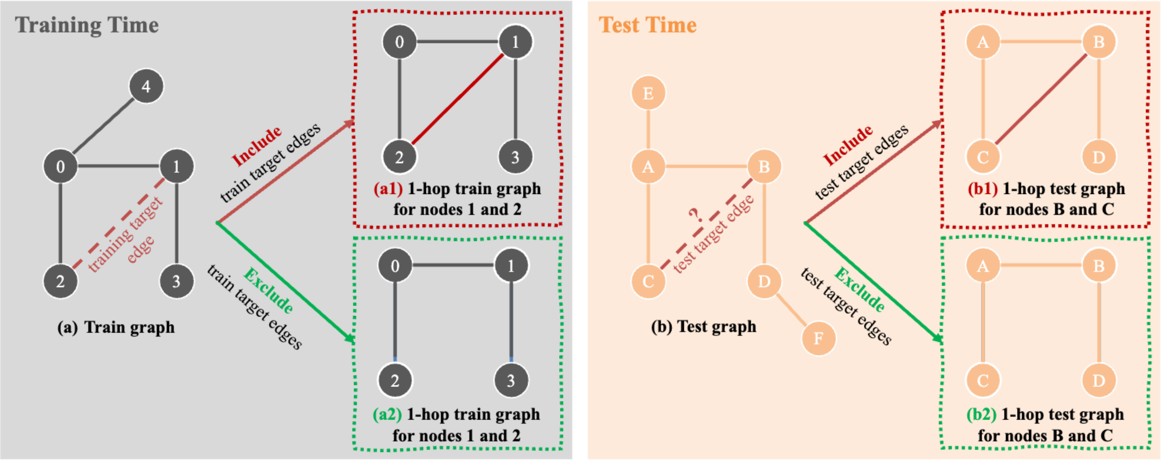
GEMS LAB

Models	SpotTarget		Baseline
	ExcludeValTst	ExcludeTst	ExcludeNone(Tst)
	Ogbl-Collab (H@50 ↑)		
SAGE	48.57 ± 0.74	57.61 ± 0.88	83.82 ± 0.59
MB-GCN	43.03 ± 0.50	50.53 ± 1.10	75.41 ± 0.43
GATv2	45.61 ± 0.85	54.94 ± 0.19	84.16 ± 2.62
SEAL	57.50 ± 0.31	55.16 ± 1.94	99.91 ± 0.05
	Ogbl-Citation2 (MRR ↑)		
SAGE	82.06 ± 0.06	82.28 ± 0.11	89.22 ± 0.10
MB-GCN	79.70 ± 0.25	81.25 ± 0.22	88.32 ± 0.14
GATv2	OOM	OOM	OOM
SEAL	86.75 ± 0.20	87.01 ± 0.39	97.14 ± 0.18
	USAir (AUC ↑)		
SAGE	95.97 ± 0.17	95.51 ± 0.53	99.15 ± 0.59
MB-GCN	94.00 ± 0.14	94.11 ± 0.13	98.66 ± 0.22
GATv2	95.05 ± 0.66	94.07 ± 0.21	98.96 ± 0.11
SEAL	95.36 ± 0.24	95.10 ± 0.76	97.20 ± 0.78
No Leakage?	✓	✓	✗
Deployment	✓	✓	✗

- Due to data leakage I3, using test edges causes a fake performance boost across datasets.
- In real-world deployed systems, this should always be avoided.

Jing Zhu et. al

Thanks! Questions?



Training Target Edges	Test Target Edges	Results	Issues
Include (a1)	-	✗	(I1) Overfitting
Include (a1)	Exclude (b2)	✗	(I2) Distribution shift
-	Include (b1)	✗	(I3) Leakage
Exclude (a2)	Exclude (b2)	✓	–