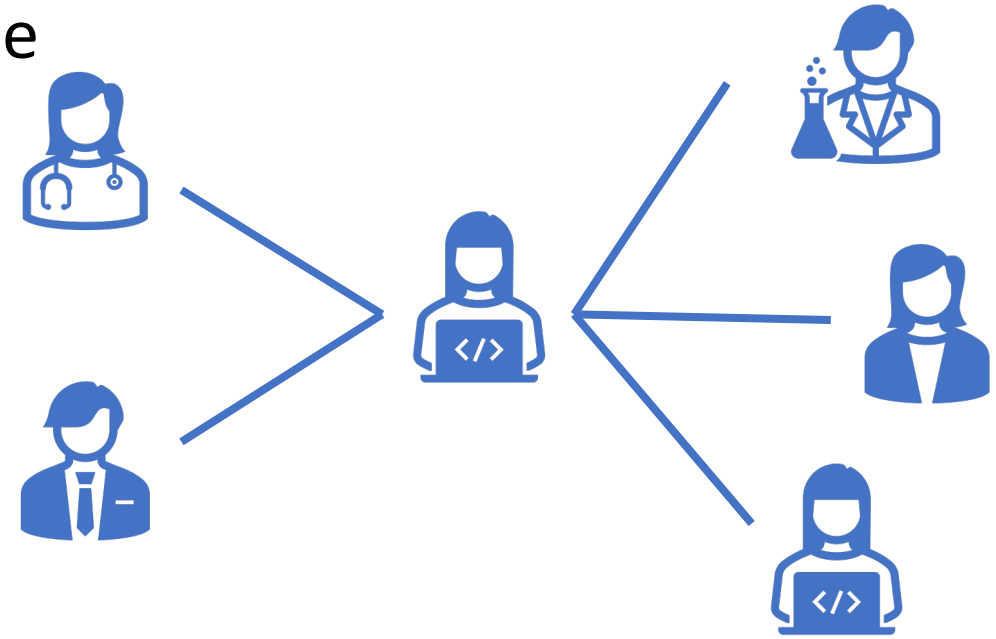
Performance-Adaptive Sampling Strategy towards Fast and Accurate Graph Neural Networks

Minji Yoon, *Carnegie Mellon University*
Théophile Gervet, *Carnegie Mellon University*
Baoxu Shi, *LinkedIn*
Sufeng Niu, *LinkedIn*
Qi He, *LinkedIn*
Jaewon Yang, *LinkedIn*

KDD2021

Background

- Graph Convolutional Networks (GCNs) becomes a powerful deep learning tool for representation learning of graph data
- Adapting GCNs to large-scale real-world graphs brings up its scalability issue



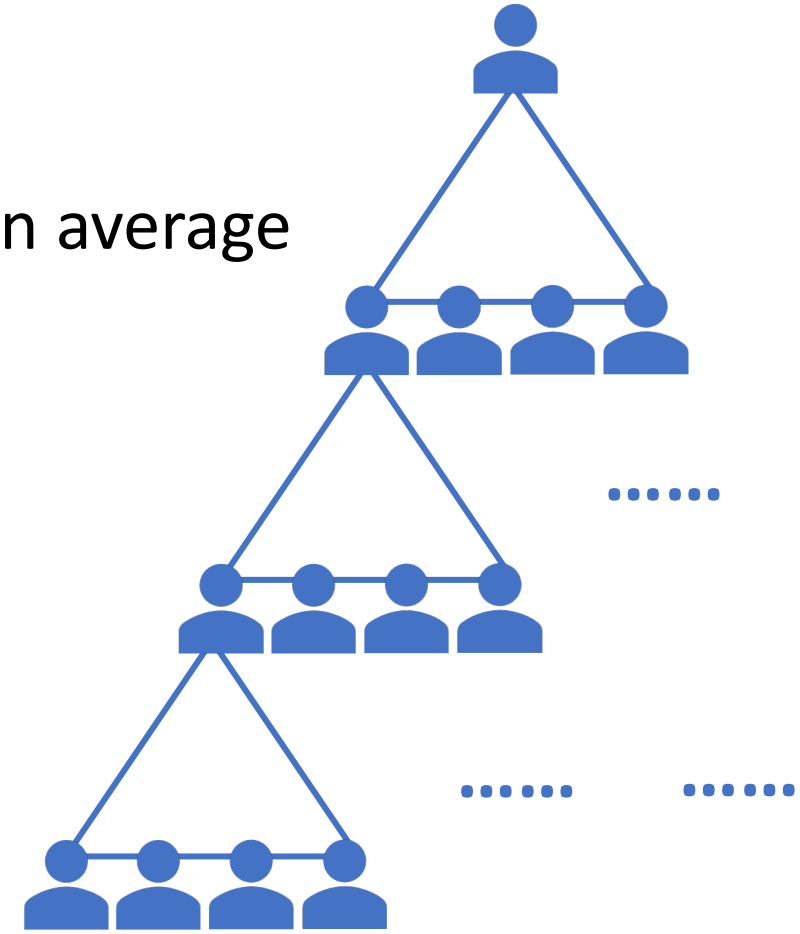
700+ million-sized member-to-member LinkedIn network

Scalability issue of GCNs

- Uncontrollable neighborhood expansion in the ***aggregation stage***

Scalability issue of GCNs

- When the average degree is d,
  L-layer GCNs access d^L neighbors per node on average

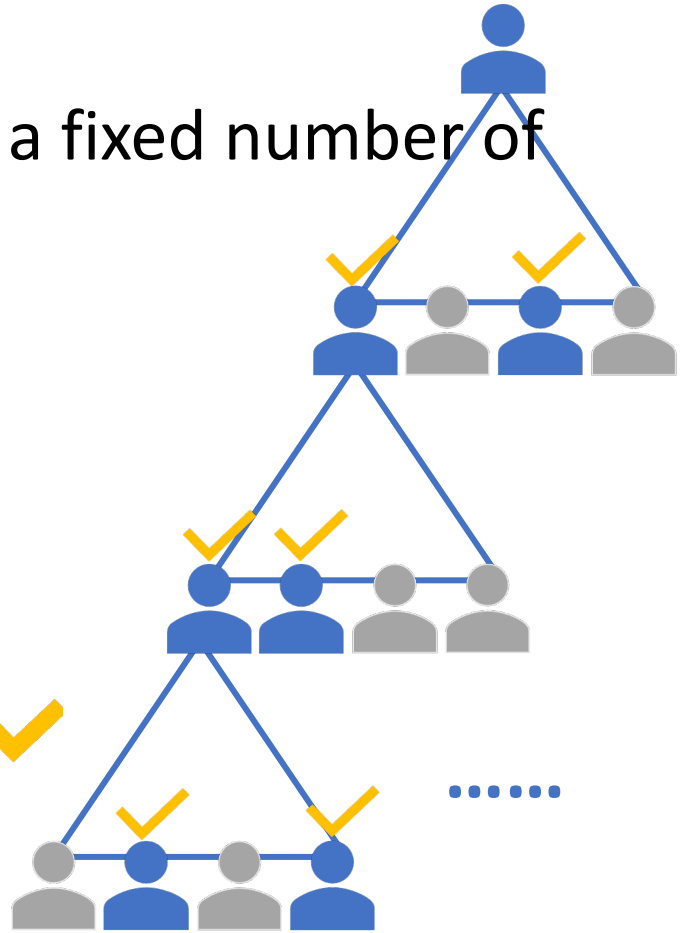


Scalability issue of GCNs

- Uncontrollable neighborhood expansion in the aggregation stage
- High computation and memory footprints

Sampling is the Solution

- Limit the neighborhood expansion by sampling a fixed number of neighbors in the aggregation operation
- Regulate the computation time and memory



Current Sampling-based GCNs

- Random Sampling: *GraphSage*[8]
- Importance Sampling: *FastGCN*[4], *LADIES*[23], *AS-GCN*[10], *GCN-BS*[14]
  - Approximate the original aggregation of the full neighborhood
  - Minimize the variance in sampling
  - Sample neighbors helpful for ***variance reduction***

[*] references could be found in our original paperc

Current Sampling-based GCNs: limitations

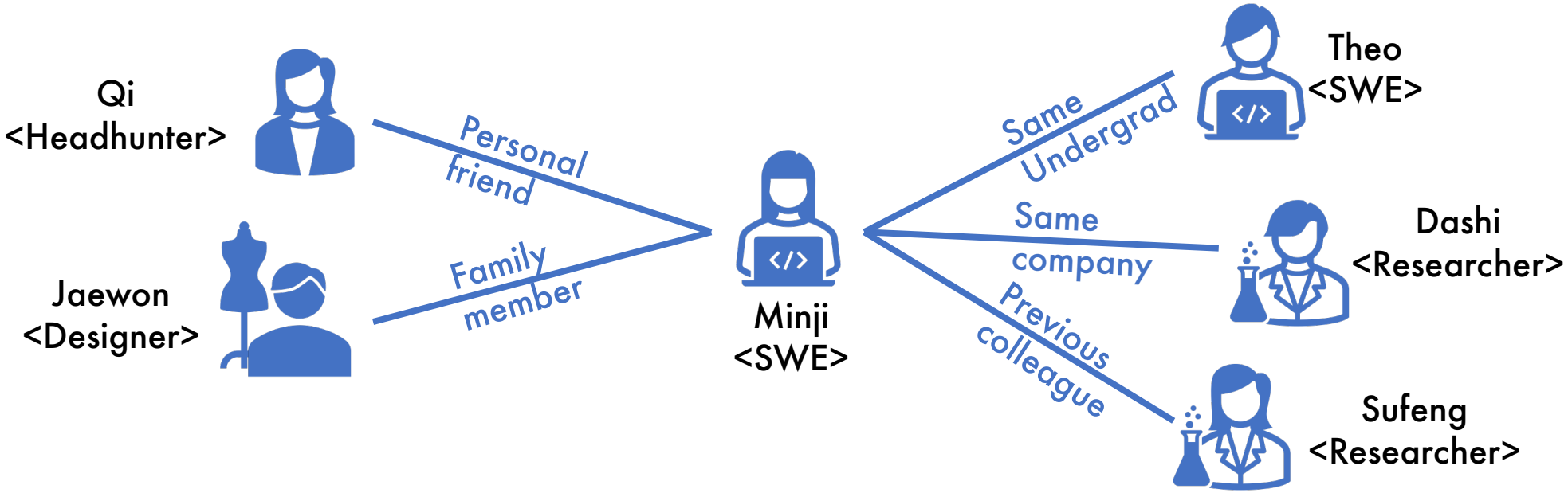
- Low accuracy
  - Sampling policy is agnostic to the performance
  - Random or for variance reduction
- Vulnerability to noise or adversarial attacks
  - Sampling policies cannot distinguish relevant neighbors from irrelevant ones
  - True neighbors from adversarially added fake neighbors

What is the optimal sampling for GCNs?

- Back to the motivation of the aggregation operation
- In GCNs, each node aggregates its neighbors' embeddings assuming that ***neighbors are informative for the target task***

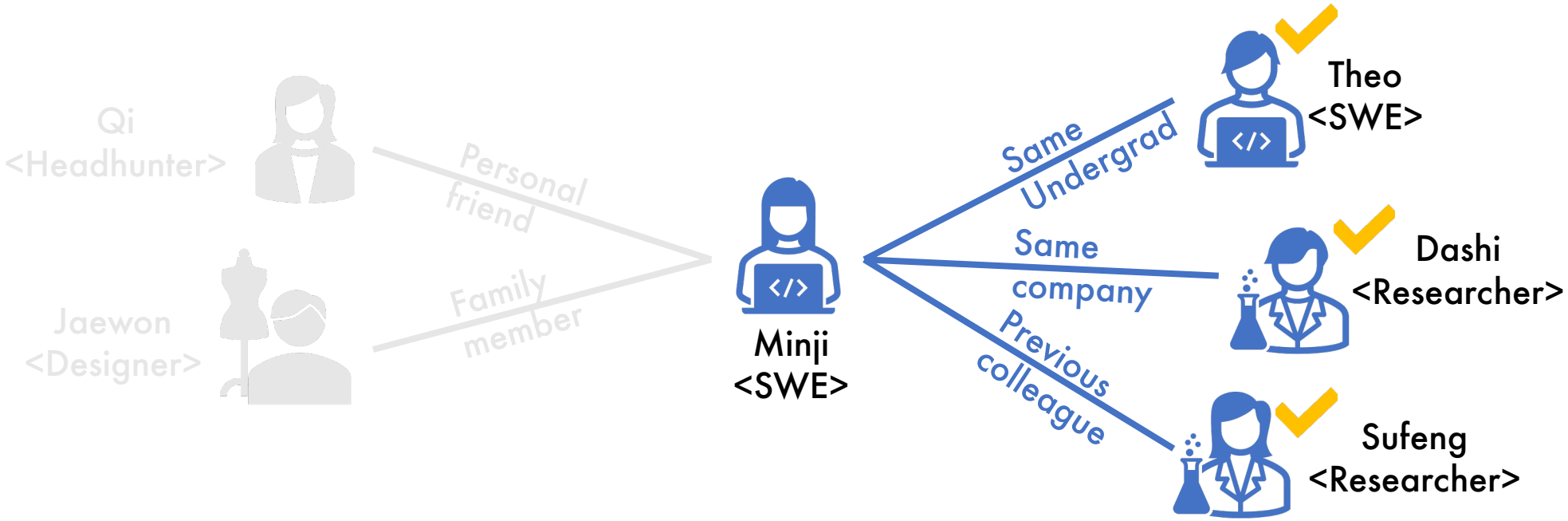
What is the optimal sampling for GCNs?

- Unfortunately, real-world graphs are noisy
- Not every neighbors are informative for the target task



What is the optimal sampling for GCNs?

- Sample neighbors informative for the target task
- We aim for a sampler that maximizes the target task's performance



Overview

1. Motivation
2. **Proposed Method**
3. Theoretical Foundation
4. Experiments
5. Conclusion

PASS: Performance-Adaptive Sampling Strategy for GCNs

GOAL: sample neighbors informative for the target task

HOW: train a sampler that directly maximizes the GCN performance

PASS: Performance-Adaptive Sampling Strategy for GCNs

GOAL: sample neighbors informative for the target task

HOW: train a sampler that directly maximizes the GCN performance

We are going to introduce..

1. **Learnable sampling policy function**
2. How to learn the parameters of the sampling policy

Sampling policy

- Parameterized sampling policy $q^{(l)}(j|i)$
  - Estimates the probability of sampling node v_j given node v_i at the l-th layer
  - Importance sampling + Random sampling

Sampling policy

- Parameterized sampling policy $q^{(l)}(j|i)$
  - Estimates the probability of sampling node v_j given node v_i at the l-th layer
  - ***Importance sampling*** + Random sampling

$$q_{imp}^{(l)}(j|i) = \left(W_s \cdot h_i^{(l)}\right) \cdot \left(W_s \cdot h_j^{(l)}\right)$$

  - W_s : Learnable transformation matrix
  - $h_i^{(l)}$: Hidden embedding of node v_i at the l-th layer

Sampling policy

- Parameterized sampling policy $q^{(l)}(j|i)$
  - Estimates the probability of sampling node v_j given node v_i at the l-th layer.
  - Importance sampling + ***Random sampling***

$$q_{imp}^{(l)}(j|i) = \left(W_s \cdot h_i^{(l)}\right) \cdot \left(W_s \cdot h_j^{(l)}\right)$$
$$q_{rand}^{(l)}(j|i) = \frac{1}{N(i)}$$

  - $N(i)$: Degree of node v_i

Sampling policy

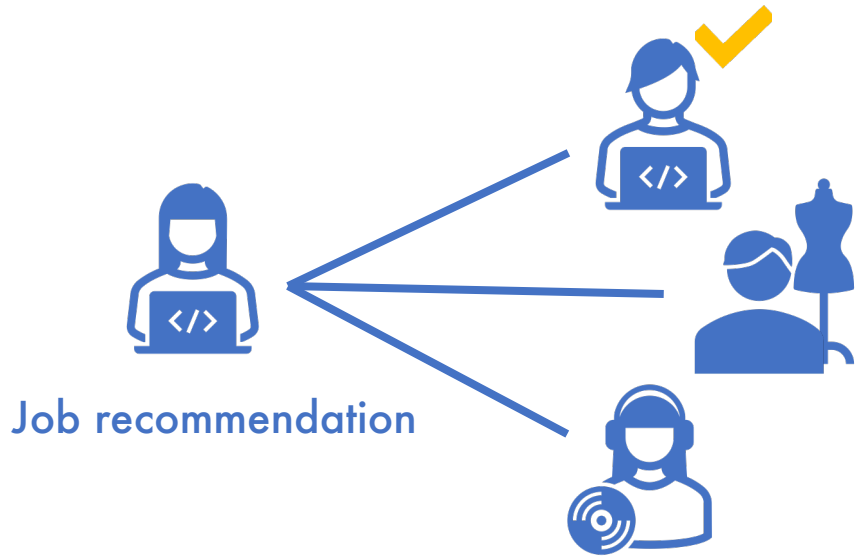
- Parameterized sampling policy $q^{(l)}(j|i)$
  - Estimates the probability of sampling node v_j given node v_i at the l-th layer.
  - Importance sampling + Random sampling

$$q_{imp}^{(l)}(j|i) = \left(W_s \cdot h_i^{(l)}\right) \cdot \left(W_s \cdot h_j^{(l)}\right)$$
$$q_{rand}^{(l)}(j|i) = \frac{1}{N(i)}$$
$$\tilde{q}^{(l)}(j|i) = a_s \cdot [q_{imp}^{(l)}(j|i),\ q_{rand}^{(l)}(j|i)]$$
$$q^{(l)}(j|i) = \tilde{q}^{(l)}(j|i) \,/\, \sum_{k=1}^{N(i)} \tilde{q}^{(l)}(k|i)$$

  - a_s : **Learnable attention vector**

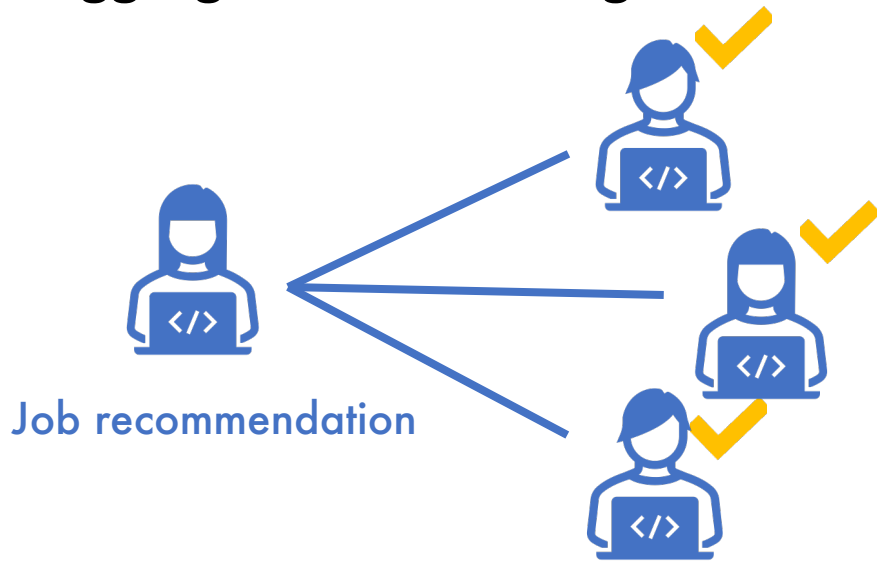
Sampling policy

- Combination of importance and random sampling
- Case 1: a graph is noisy
  - **Importance sampling** helps differentiate related and unrelated neighbors



Sampling policy

- Combination of importance and random sampling
- Case 2: a graph is well-clustered
  - Nodes are connected with all informative neighbors.
  - **Randomness** helps aggregate diverse neighbors and prevents the GCN from overfitting



Sampling policy

- Combination of importance and random sampling
- Generalize better across various graphs
- Learn which sampling methodology is more effective
  - Attention of sampling a_s

$$\tilde{q}^{(l)}(j|i) = \boldsymbol{a_s} \cdot [q_{imp}^{(l)}(j|i), \quad q_{rand}^{(l)}(j|i)]$$

Sampling policy

- Parameter sharing
  - Share the parameters (W_s , a_s) across all edges
  - Generalize and prevent the sampling policy from overfitting to the training set

PASS: Performance-Adaptive Sampling Strategy for GCNs

GOAL: sample neighbors informative for the target task

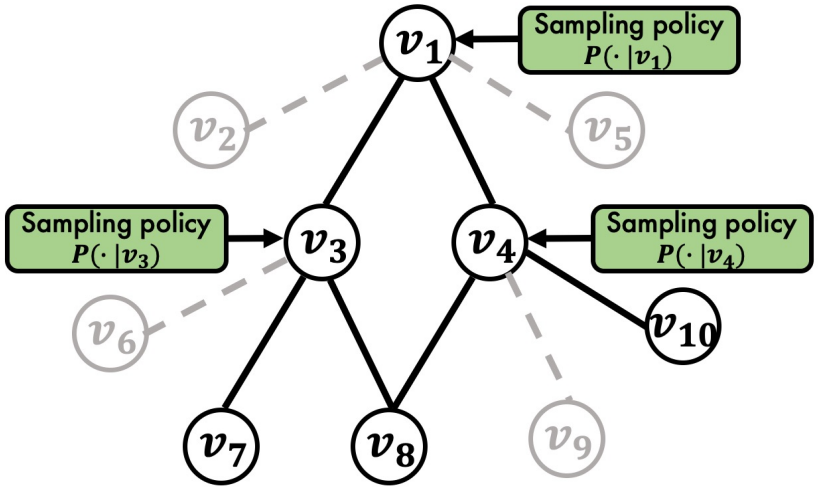
HOW: train a sampler that directly maximizes the GCN performance

We are going to introduce..

1. Learnable sampling policy function
2. **How to learn the parameters of the sampling policy**

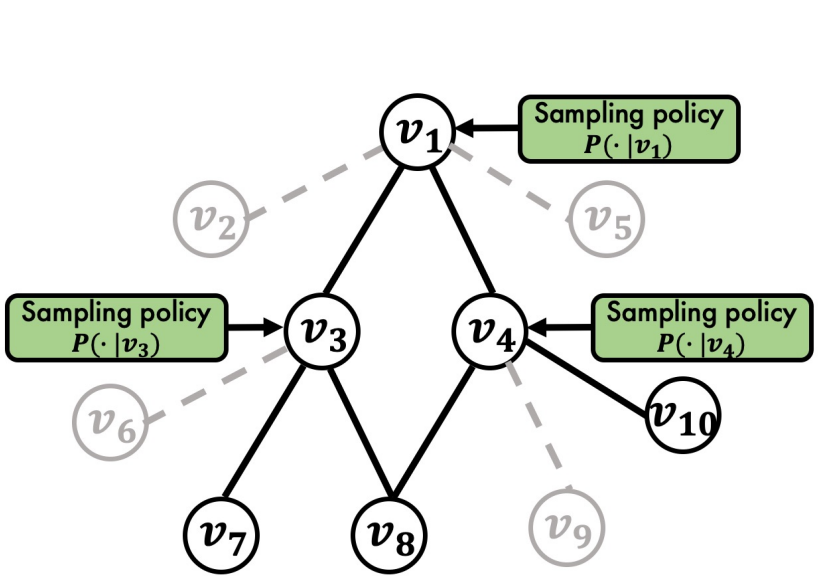
Training the Sampling policy

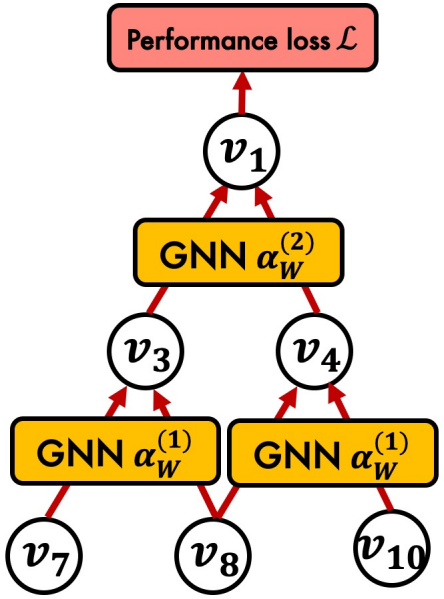
STEP1: generate a computation graph using the sampling policy $q^{(l)}(j|i)$ in a top-down manner ($l: L \rightarrow 1$)



Training the Sampling policy

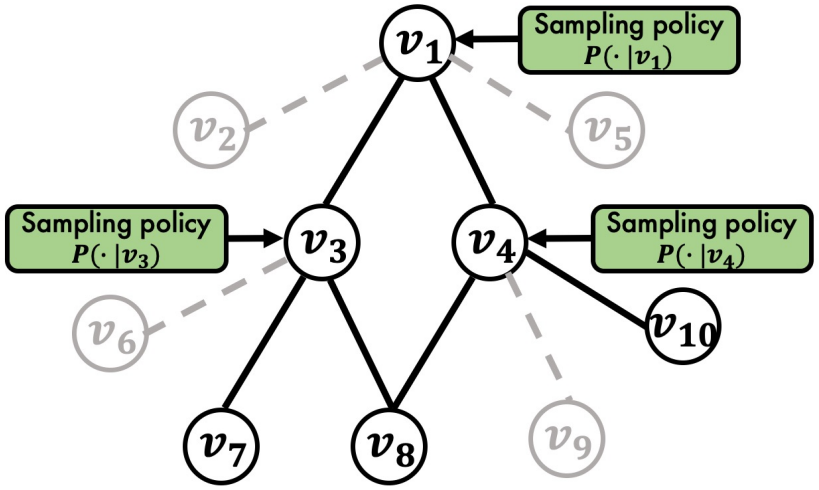
STEP2: after acquiring the computation graph, we do feedforward propagation in a bottom-up manner ($l: 1 \rightarrow L$)

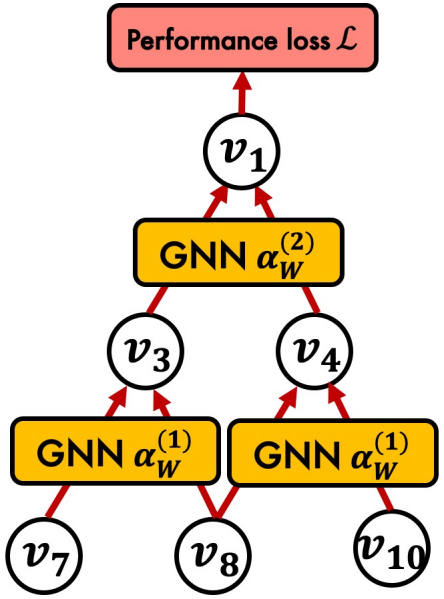


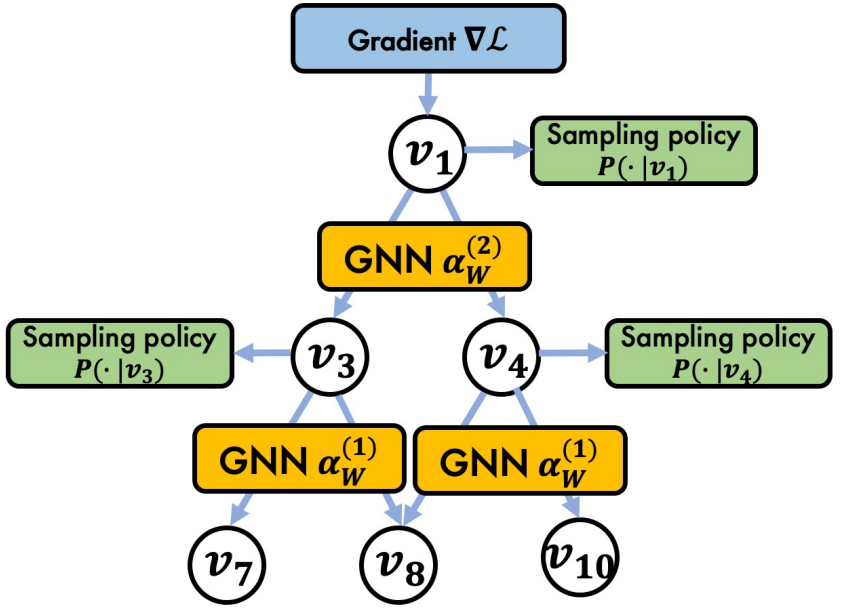


Training the Sampling policy

STEP3: in the backpropagation phase, we update parameters using gradients of the loss in a top-down manner ($l: L \rightarrow 1$)



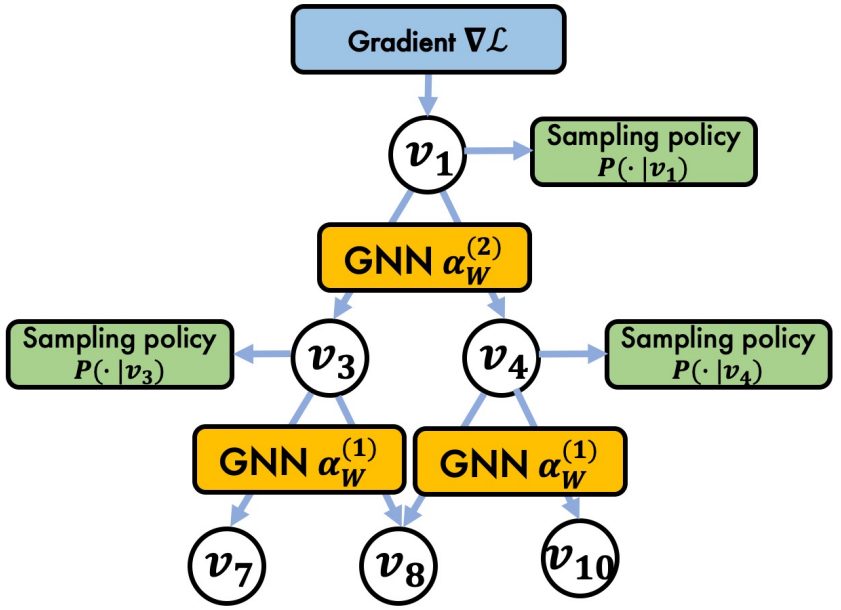




Training the Sampling policy

STEP3: In the backpropagation phase, we update parameters using gradients of the loss in a top-down manner ($l: L \rightarrow 1$)

Update the parameters of both the GCN and sampling policy using gradients of the performance loss



Training the Sampling policy

- When θ denotes parameters (W_s , a_s) in the sampling policy $q_\theta^{(l)}$, the sampling operation is presented as:

$$h_i^{(l+1)} = \alpha_{W^{(l)}}\left(\mathbb{E}_{j\sim q_\theta^{(l)}(j|i)}\left[h_j^{(l)}\right]\right), l = 0, \cdots, L-1$$

  - $\alpha_{W^{(l)}}$: Nonlinear unit + transformation matrix at l–th layer in GCNs

Training the Sampling policy

- When θ denotes parameters (W_s , a_s) in the sampling policy $q_\theta^{(l)}$, the sampling operation is presented as:

$$h_i^{(l+1)} = \alpha_{W^{(l)}} \left(\mathbb{E}_{j \sim q_\theta^{(l)}(j|i)} \left[h_j^{(l)} \right] \right), l = 0, \cdots, L-1$$

  - $\alpha_{W^{(l)}}$: Nonlinear unit + transformation matrix at l–th layer in GCNs

How does $\nabla_\theta \mathcal{L}$ pass through the non-differentiable sampling operation $\mathbb{E}_{q_\theta^{(l)}}[\cdot]$ to update our sampling policy $q_\theta^{(l)}$?

Training the Sampling policy

THEOREM 4.1. *Given the loss $\mathcal{L}$ and the hidden embedding $h_i^{(l)}$ of node v_i at the l-th layer, the gradient of $\mathcal{L}$ w.r.t. the parameter θ of the sampling policy $q_\theta^{(l)}(j|i)$ is computed as follows:*

$$\nabla_\theta \mathcal{L} = \frac{d\mathcal{L}}{dh_i^{(l+1)}} \alpha_{W^{(l)}} \mathbb{E}_{j \sim q_\theta^{(l)}(j|i)} [\nabla_\theta \log q_\theta^{(l)}(j|i) h_j^{(l)}]$$

- **Log derivative trick:** widely used in reinforcement learning to compute gradients of stochastic policies

Training the Sampling policy

THEOREM 4.1. *Given the loss $\mathcal{L}$ and the hidden embedding $h_i^{(l)}$ of node v_i at the l-th layer, the gradient of $\mathcal{L}$ w.r.t. the parameter θ of the sampling policy $q_\theta^{(l)}(j|i)$ is computed as follows:*

$$\nabla_\theta \mathcal{L} = \frac{d\mathcal{L}}{dh_i^{(l+1)}} \alpha_{W^{(l)}} \mathbb{E}_{j \sim q_\theta^{(l)}(j|i)} [\nabla_\theta \log q_\theta^{(l)}(j|i) h_j^{(l)}]$$

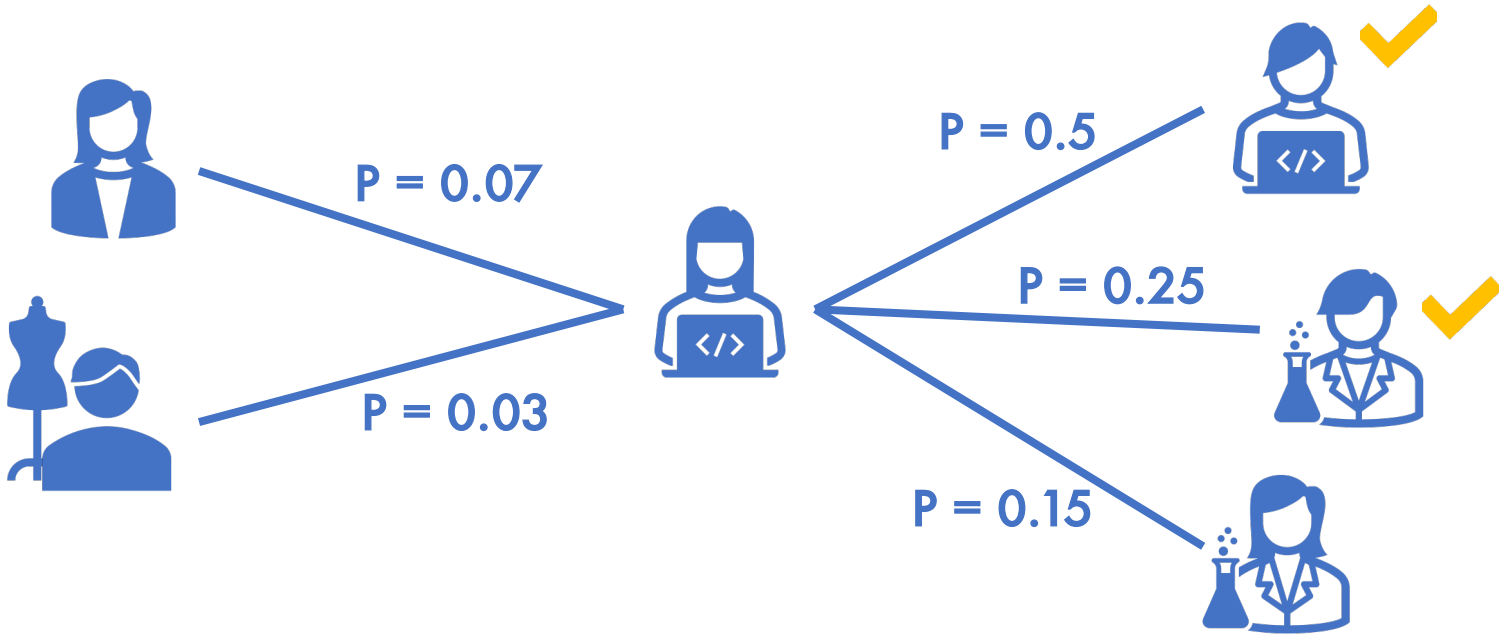
- **Log derivative trick:** widely used in reinforcement learning to compute gradients of stochastic policies
- The gradients of GCN performance loss optimize the sampling policy directly for the GCN performance

Overview

1. Motivation
2. Proposed Method
3. **Theoretical Foundation**
4. Experiments
5. Conclusion

Theoretical Foundation

- How does **PASS** learn whether a neighbor is informative for the target task *from gradients of the performance loss*?
- Why does **PASS** assign *a certain sampling probability* to the neighbor?



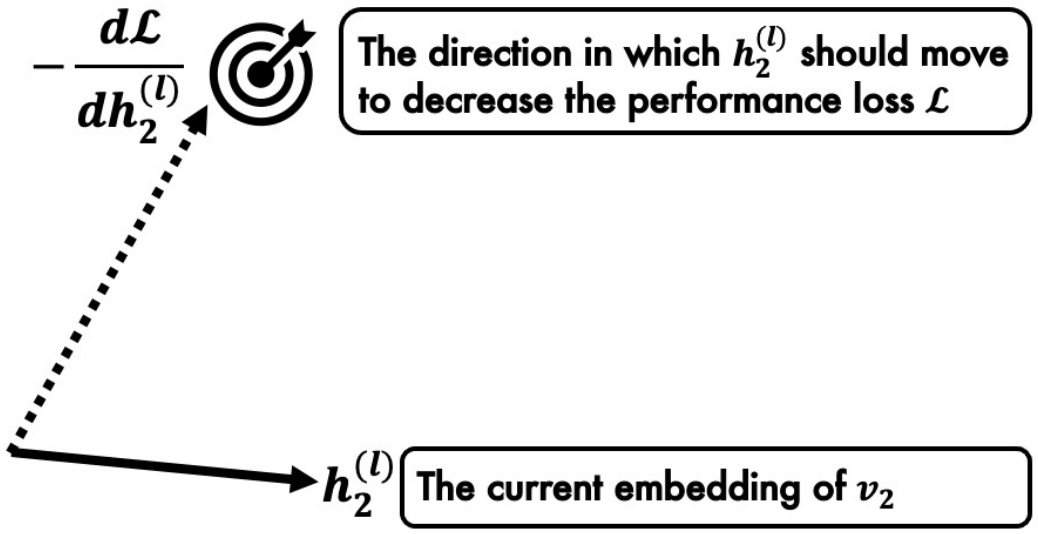
Theoretical Foundation

- **PASS** decides a neighbor node v_j is informative when its embedding $\boldsymbol{h_j^{(l)}}$ ***is aligned with the gradient*** $\boldsymbol{-d\mathcal{L}/dh_i^{(l)}}$
- **PASS** increases a sampling probability of node v_j ***in proportion to the dot product of*** $\boldsymbol{-d\mathcal{L}/dh_i^{(l)}}$ ***and its embedding*** $\boldsymbol{h_j^{(l)}}$

THEOREM 5.1. *Given a source node v_i and its neighbor node v_j, PASS increases a sampling probability $q^{(l)}(j|i)$ in proportion to the dot product of $-d\mathcal{L}/dh_i^{(l)}$ and $h_j^{(l)}$.*

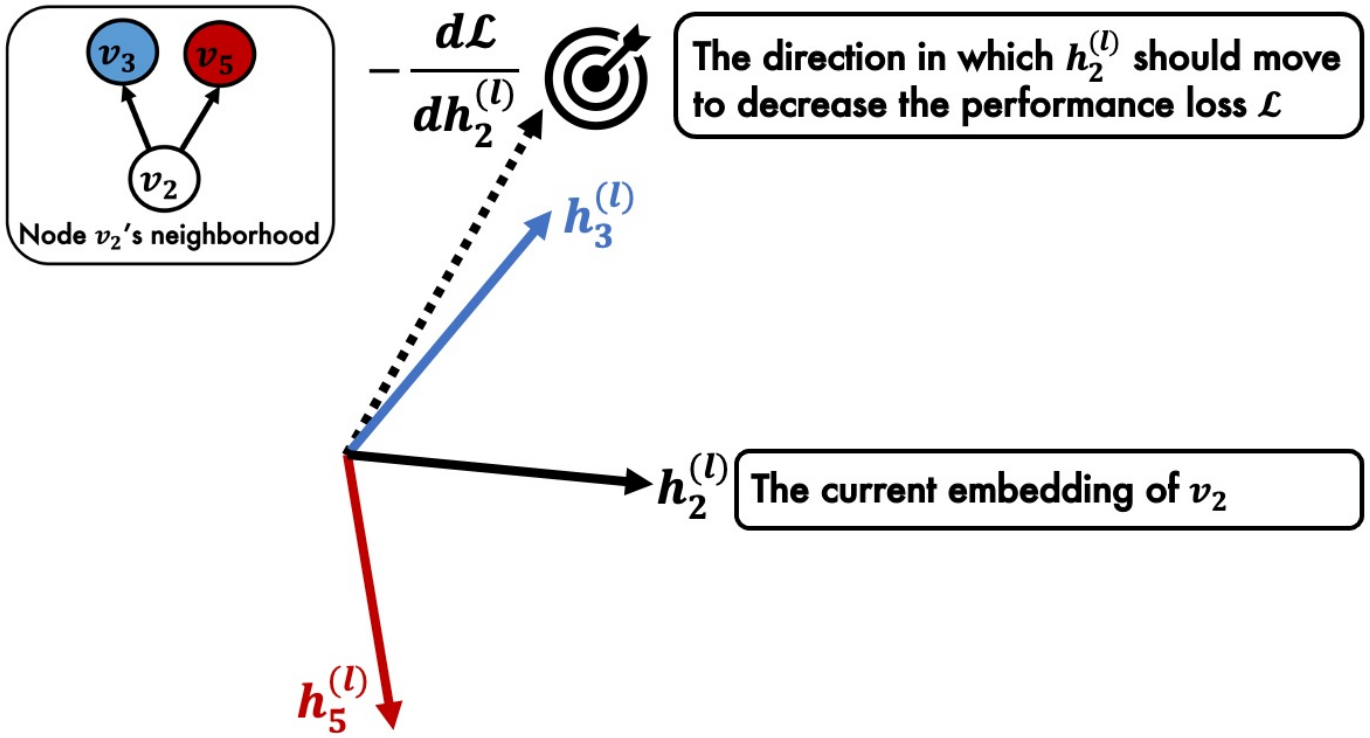
Theoretical Foundation

- Node embeddings $h_i^{(l)}$ is moved in the direction that minimizes the performance loss $\mathcal{L}$ during back-propagation step



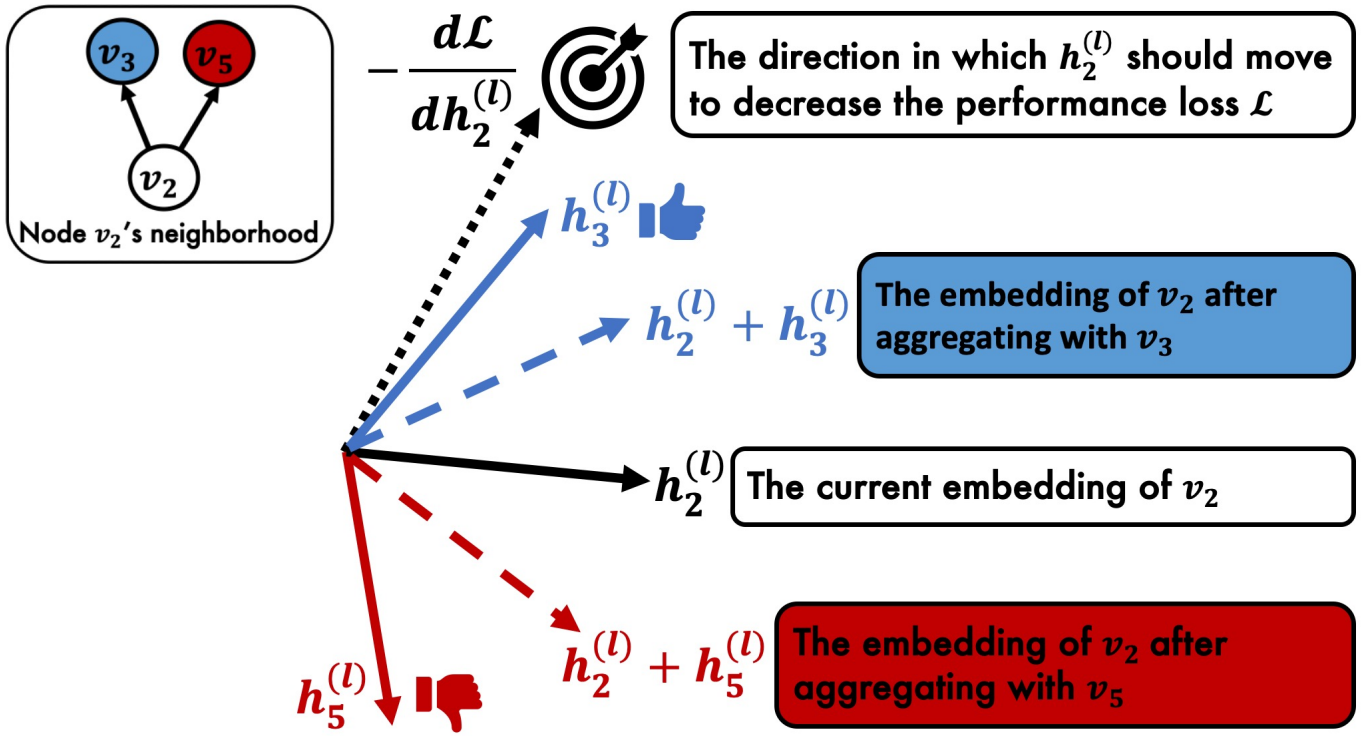
Theoretical Foundation

- Node embeddings $h_i^{(l)}$ is also moved by the aggregation operation



Theoretical Foundation

- Aggregating with neighbors aligned with the gradient $-\nabla\mathcal{L}$ helps the embedding $h_i^{(l)}$ move in the direction that reduces the loss $\mathcal{L}$



Yoon et al., Performance-Adaptive Sampling Strategy towards Fast and Accurate Graph Neural Networks, KDD'21

Theoretical Foundation

This analysis allows us to understand…

- Functionality of aggregation operations
  - Move node embeddings to reduce the performance loss
- Joint optimization of the GCN and sampling policy
  - GCNs reply only on its parameters to minimize the performance loss
  - Both parameters are updated together towards the minimum loss

Overview

1. Motivation
2. Proposed Method
3. Theoretical Foundation
4. **Experiments**
5. Conclusion

Experimental Setting

- Semi-supervised node classification tasks
- 7 public datasets and 2 LinkedIn datasets
  - 3 citation networks: Cora, Citeseer, Pubmed
  - 2 co-purchase graphs: Amazon Computers, Amazon Photo
  - 2 co-authorship graphs: MS Computer Science, MS Physics
  - 2 subsets of LinkedIn social networks

Experimental Setting

- Baselines
  - Node-wise sampler: GraghSage, GCN-BS
  - Layer-wise sampler: FastGCN, AS-GCN
  - Attention method: GAT
  - (PASS is a node-wise sampler)
- Unified time complexity bound
  - With the batch size set to 64
  - Layer-wise samplers sample 64 nodes per layer
  - Node-wise samplers sample one neighbor per node, thus sampling 64 nodes per layer in total

Experimental Setting

- **Evaluation with Sampling**
  - Prohibitive time and memory costs from the full neighborhood expansion are also issues during testing
  - Sample both during training and testing
  - Significant drop in accuracy for certain baselines, especially layer-wise samplers

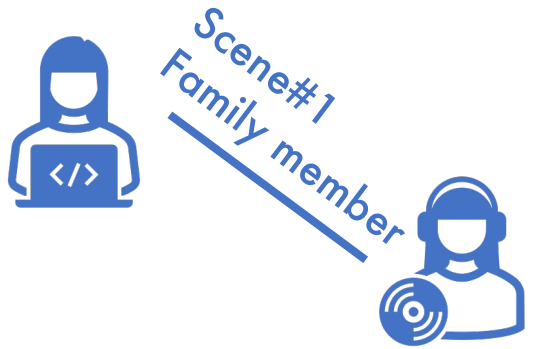
Q1. Effectiveness

Method	Cora	Citeseer	Pubmed	AmazonC	AmazonP	MsCS	MsPhysics	LnkIndustry	LnkTitle
FastGCN	0.582	0.496	0.569	0.480	0.542	0.520	0.638	-4.2pp	-2.0pp
AS-GCN	0.462	0.387	0.502	0.419	0.480	0.403	0.516	-7.1pp	-0.6pp
GraphSage	0.788	0.698	0.792	0.707	0.787	0.766	0.875	0.0pp	0.0pp
GCN-BS	0.788	0.693	0.809	0.736	0.800	0.780	0.887	1.8pp	0.7pp
PASS	**0.821**	**0.715**	**0.858**	**0.757**	**0.855**	**0.884**	**0.934**	**10.2pp**	**1.3pp**

* Results on LinkedIn datasets are presented in percentage point (pp) w.r.t GraphSage

- **PASS** shows the highest accuracy among all baselines across all datasets
- Layer-wise methods (FastGCN, AS-GCN) show lower accuracy than node-wise methods (GraphSage, GCN-BS, PASS)

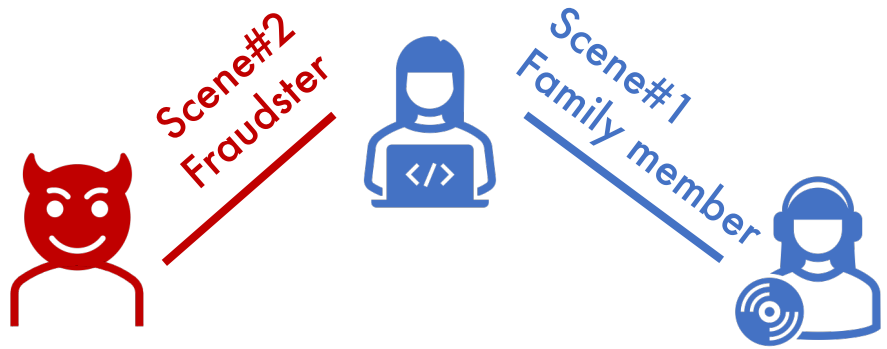
Q2. Robustness



Two different noise scenarios

1. Fake connections among existing nodes
   - Connections made by mistake or unfit for purpose
   - e.g., Connections between family members in a job search platform

Q2. Robustness

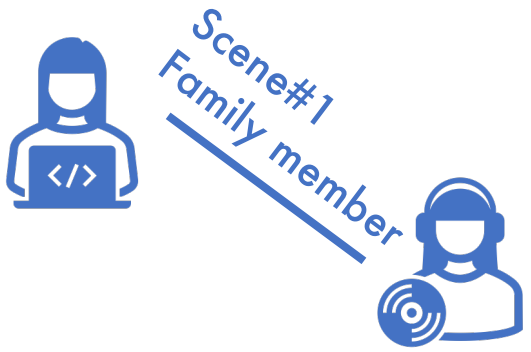


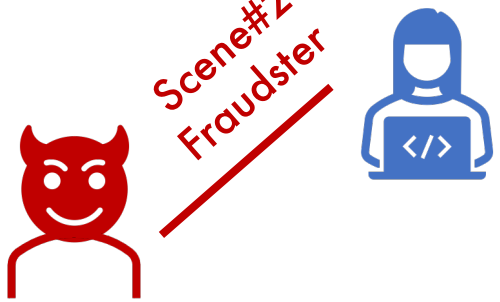
Two different noise scenarios

1. Fake connections among existing nodes
   - Connections made by mistake or unfit for purpose
   - e.g., Connections between family members in a job search platform
2. Fake neighbors with random feature vectors
   - Random connections to existing nodes
   - e.g.. Fake accounts with random attributes used for fraudulent activities

For each node, we generate five true neighbors and five fake neighbors

Q2. Robustness

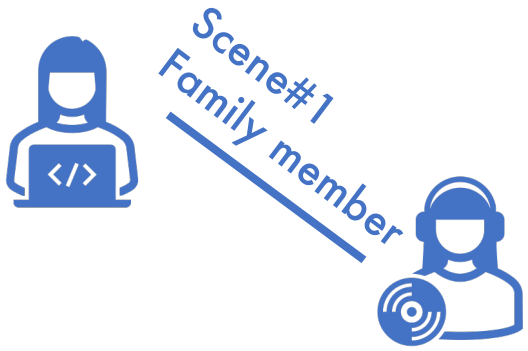




	Fake connections among existing nodes					
Method	**Cora**	**Citeseer**	**Pubmed**	**AmazonC**	**AmazonP**	**MsCS**
FastGCN	0.293	0.254	0.416	0.300	0.307	0.292
AS-GCN	0.229	0.171	0.334	0.206	0.167	0.176
GraphSage	0.312	0.261	0.439	0.376	0.306	0.262
GCN-BS	0.320	0.265	0.457	0.387	0.305	0.264
PASS	**0.658**	**0.603**	**0.811**	**0.669**	**0.698**	**0.822**

	Fake neighbors with random feature vectors					
Method	**Cora**	**Citeseer**	**Pubmed**	**AmazonC**	**AmazonP**	**MsCS**
FastGCN	0.597	0.513	0.614	0.502	0.566	0.563
AS-GCN	0.233	0.152	0.379	0.271	0.169	0.252
GraphSage	0.282	0.269	0.459	0.264	0.264	0.248
GCN-BS	0.571	0.493	0.681	0.639	0.686	0.622
PASS	**0.722**	**0.681**	**0.761**	**0.672**	**0.783**	**0.667**

Q2. Robustness

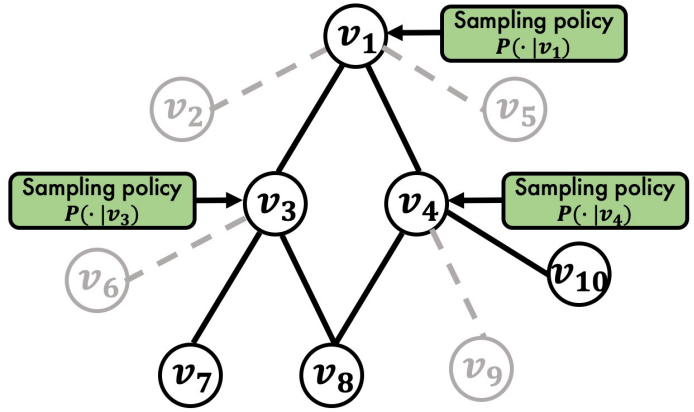


[Future work]
We can extend our sampler to detect fake page or fake account

Method	Fake connections among existing nodes					
	Cora	Citeseer	Pubmed	AmazonC	AmazonP	MsCS
FastGCN	0.293	0.254	0.416	0.300		
AS-GCN	0.229	0.171	0.334			
GraphSage	0.312	0.261				
GCN-BS	0.320					0.264
PASS					0.698	0.822

	with random feature vectors					
		Pubmed	AmazonC	AmazonP	MsCS	
	0.597	0.513	0.614	0.502	0.566	0.563
AS-GCN	0.233	0.152	0.379	0.271	0.169	0.252
GraphSage	0.282	0.269	0.459	0.264	0.264	0.248
GCN-BS	0.571	0.493	0.681	0.639	0.686	0.622
PASS	**0.722**	**0.681**	**0.761**	**0.672**	**0.783**	**0.667**

Q3. Comparison with GATs



- **PASS** (Ours)
  - Sample all neighbors using their feature vectors
  - Generate a small computation graph before forward/backward propagation on GCNs
- **GAT**
  - Hold the whole graph while training GCNs
  - Matrix multiplications with large-sized adjacency matrices

Q3. Comparison with GATs

- Batch size: 64
- Sampling number: 1
- Average degree: 15
- Number of layers: 3
- **PASS** (Ours)
  - #nodes: 64 + 64 + 64 = 192
  - Adjacency matrix: $O(10^4)$
- **GAT**
  - #nodes: 64 + (64 * 15) + (64 * 15 * 15) = 15,424
  - Adjacency matrix: $O(10^8)$

Q3. Comparison with GATs

	Accuracy			Training time (s)			Test time (s)		
Dataset	**GATs**	**PASS (1)**	**PASS (5)**	**GATs**	**PASS (1)**	**PASS (5)**	**GATs**	**PASS (1)**	**PASS (5)**
Cora	**0.850**	0.821	**0.847**	189.670	9.459	7.226	0.122	0.022	0.033
Citeseer	**0.744**	0.715	0.735	404.904	13.962	13.225	0.175	0.043	0.069
Pubmed	-	0.858	**0.871**	-	87.660	94.918	-	0.612	1.510
AmazonC	-	0.757	**0.886**	-	52.060	184.522	-	0.256	1.218
AmazonP	0.905	0.855	**0.944**	1869.690	30.060	68.134	0.709	0.094	0.338
MS CS	-	0.884	**0.918**	-	101.840	142.099	-	0.811	3.113
MS Physics	-	0.934	**0.952**	-	439.378	507.816	-	4.162	8.445

* PASS (1)/(5) samples 1/5 neighbors per node, trading-off speed for accuracy

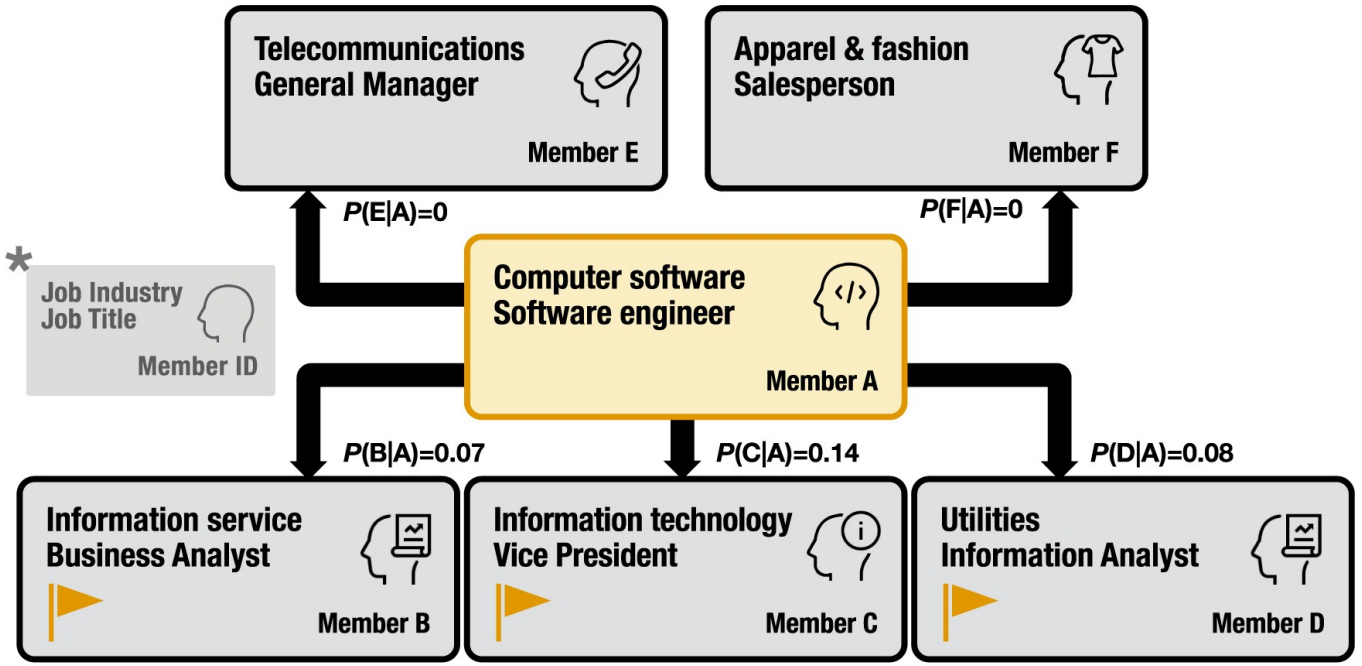
- **PASS** is scalable across all datasets
- **GAT** runs out of memory on 4 out of 7 datasets
- **PASS-5** shows comparable or higher accuracy as **GAT** while maintaining *shorter training and test times*

Overview

1. Motivation
2. Proposed Method
3. Theoretical Foundation
4. Experiments
5. **Conclusion**

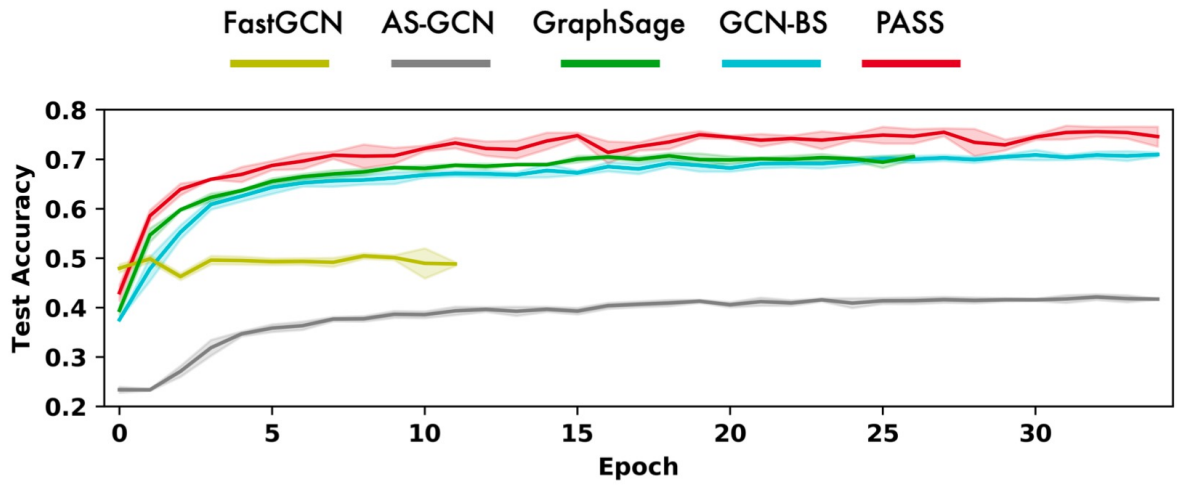
Conclusion

- **Performance-adaptiveness**
  - PASS learns to sample neighbors informative for the task performance
- **Effectiveness**
  - PASS outperforms state-of-the-art samplers, being 10.4% more accurate
- **Robustness**
  - PASS shows up to 53.1% higher accuracy in the presence of adversarial attacks
- **Theoretical foundation**
  - PASS presents how it learns whether a neighbor is informative

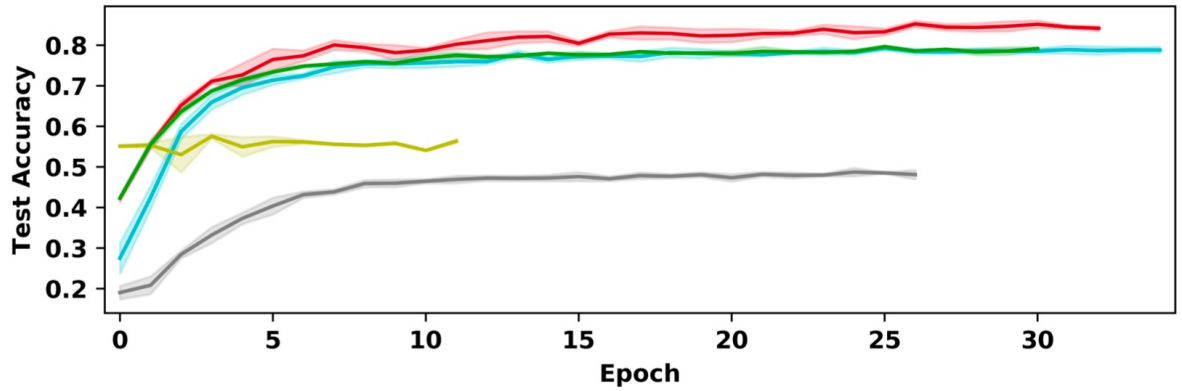

Telecommunications
General Manager
Member E
Apparel & fashion
Salesperson
Member F
P(E|A)=0
P(F|A)=0
*
Job Industry
Job Title
Member ID
Computer software
Software engineer
Member A
P(B|A)=0.07
P(C|A)=0.14
P(D|A)=0.08
Information service
Business Analyst
Member B
Information technology
Vice President
Member C
Utilities
Information Analyst
Member D

Thank you

Appendix



(a) Amazon Computer



(b) Amazon Photo