GraphSAINT: Graph Sampling Based Inductive Learning Method

Hanqing Zeng*[,1], Hongkuan Zhou*[,1], Ajitesh Srivastava[1], Rajgopal Kannan[2], Viktor Prasanna[1]

1. University of Southern California
2. US Army Research Lab

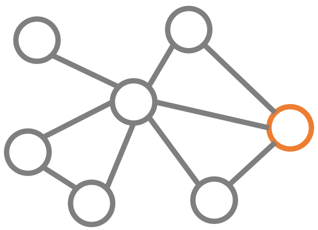
ICLR, April 2020

https://github.com/GraphSAINT/

fpga.usc.edu

Background: Graph Representation Learning

What it does: Embed each graph node into a low-dimensional vector



GNN

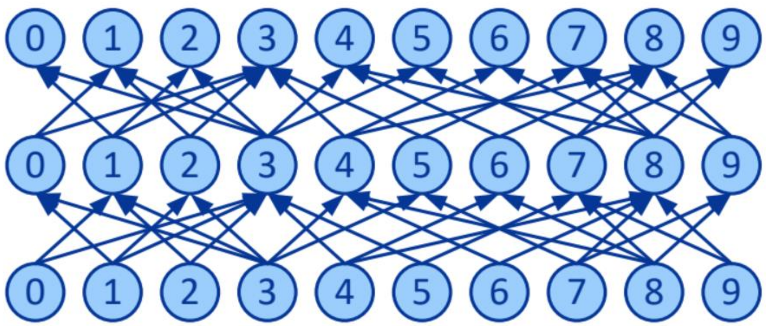
$$x = \begin{bmatrix} 1.2 \\ -2 \\ 5.3 \end{bmatrix}$$

Clustering

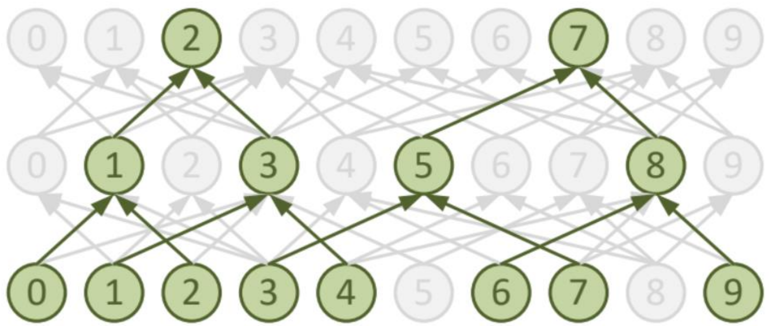
Link prediction

Node classification

Why it is challenging: Num. of multi-hop neighbors explode



GNN on the full graph



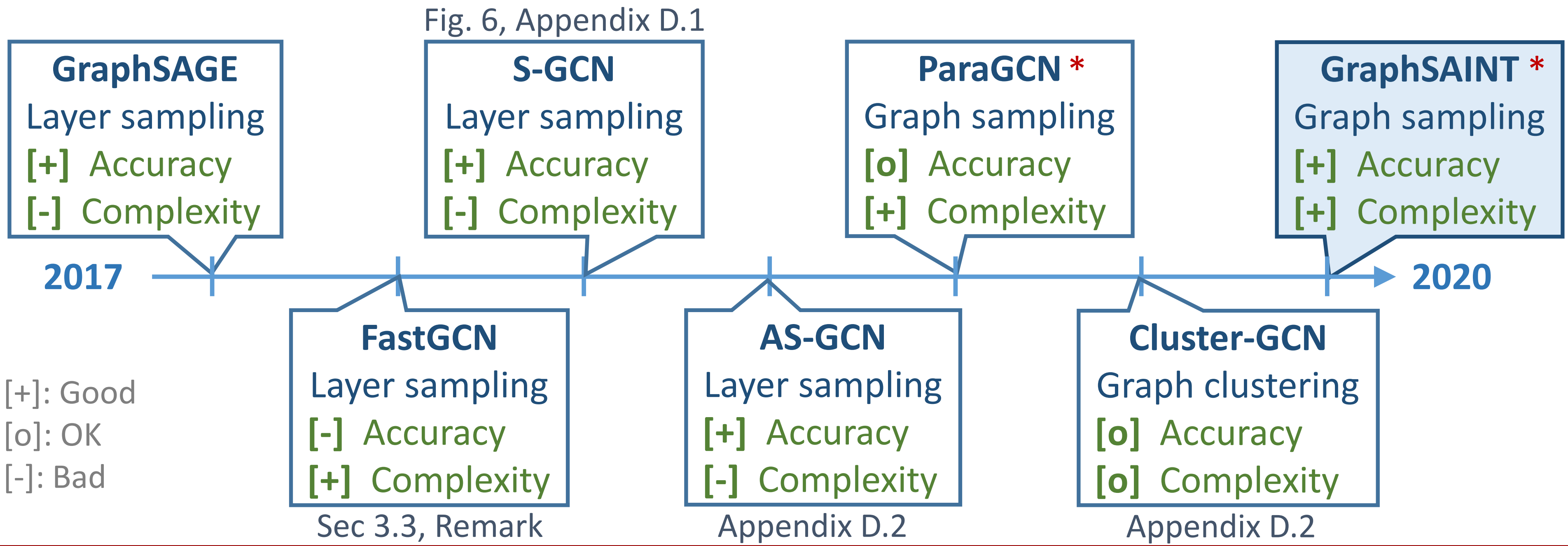
Sampled GNN of a minibatch

Background: Graph Representation Learning

Low training efficiency is now a serious problem in GNN deployment!!

- PinSAGE 2-layer GCN + 7.5billion nodes = **3 days on 16-GPU cluster**
- Deep GNNs Training cost grows **exponentially** with depth

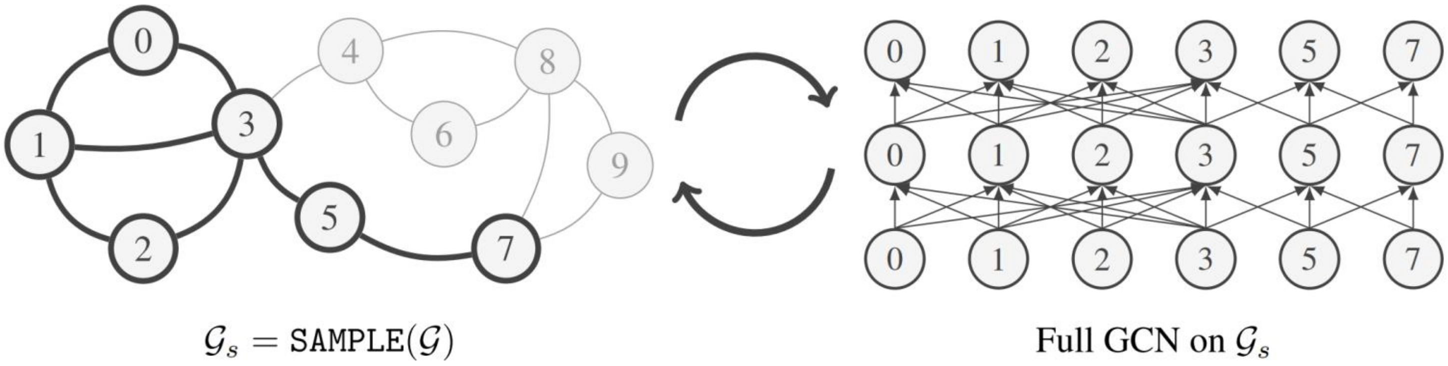
Overview of state-of-the-art minibatch training methods (ref. on slide 11)



Method: Minibatch Construction

Sample a small subgraph, then build a *complete* GNN

- Constant neighborhood size
- Not an I.I.D. data sampler → *Bias elimination*
- Need to preserve connectivity → *Variance reduction*



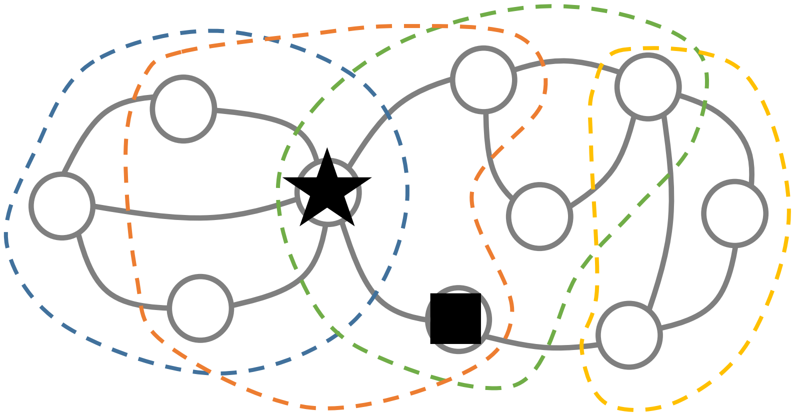
General training framework

- **Graph samplers**: for social networks, molecular graph, traffic networks, …
- **GNN architectures**: convolution, pooling, attention, skip-connection, …

Method: Bias Elimination

Why?

- Popular users will be/need to be sampled more frequently
- Need accurate embedding on everyone



How?

- **Normalize minibatch loss** by probability of sampling each <u>node</u>
- **Normalize neighbor aggregation** by probability of sampling each <u>edge</u>
- **Lightweight preprocessing** to estimate the probability

$p_\star = 3/4$

$p_{\star,\blacksquare} = 2/4$

$$\boxed{Loss_{\text{mini}} = \frac{1}{|V|}\left(\frac{1}{p_\star} Loss_\star + \cdots\right)}$$

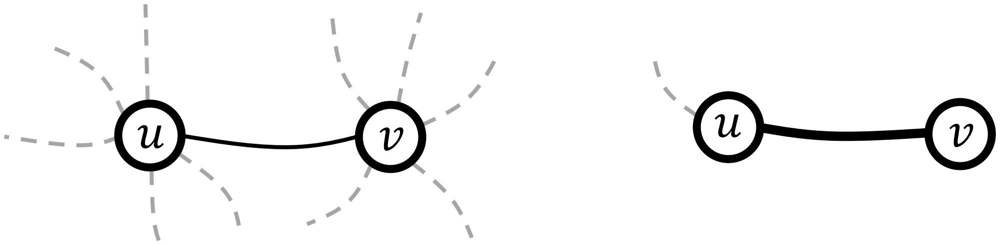
$$\boxed{\boldsymbol{x}_\star{}' = \frac{1}{d_\star}\left(\frac{1}{p_{\star,\blacksquare}} \boldsymbol{x}_\blacksquare + \cdots\right)}$$

Method: Variance Reduction

How?

- “Important” neighbors to be sampled more frequently

<u>Theorem</u> (independent edge sampling): optimal edge probability for variance minimization is $p_{u,v} \propto \frac{1}{d_u} + \frac{1}{d_v}$.



Sample “**—**” more
Sample “—” less

Extensions

- Multi-layer version of random edge sampling → Random walk sampler
- Variations of random walk sampler → Multi-dimensional RW, etc.

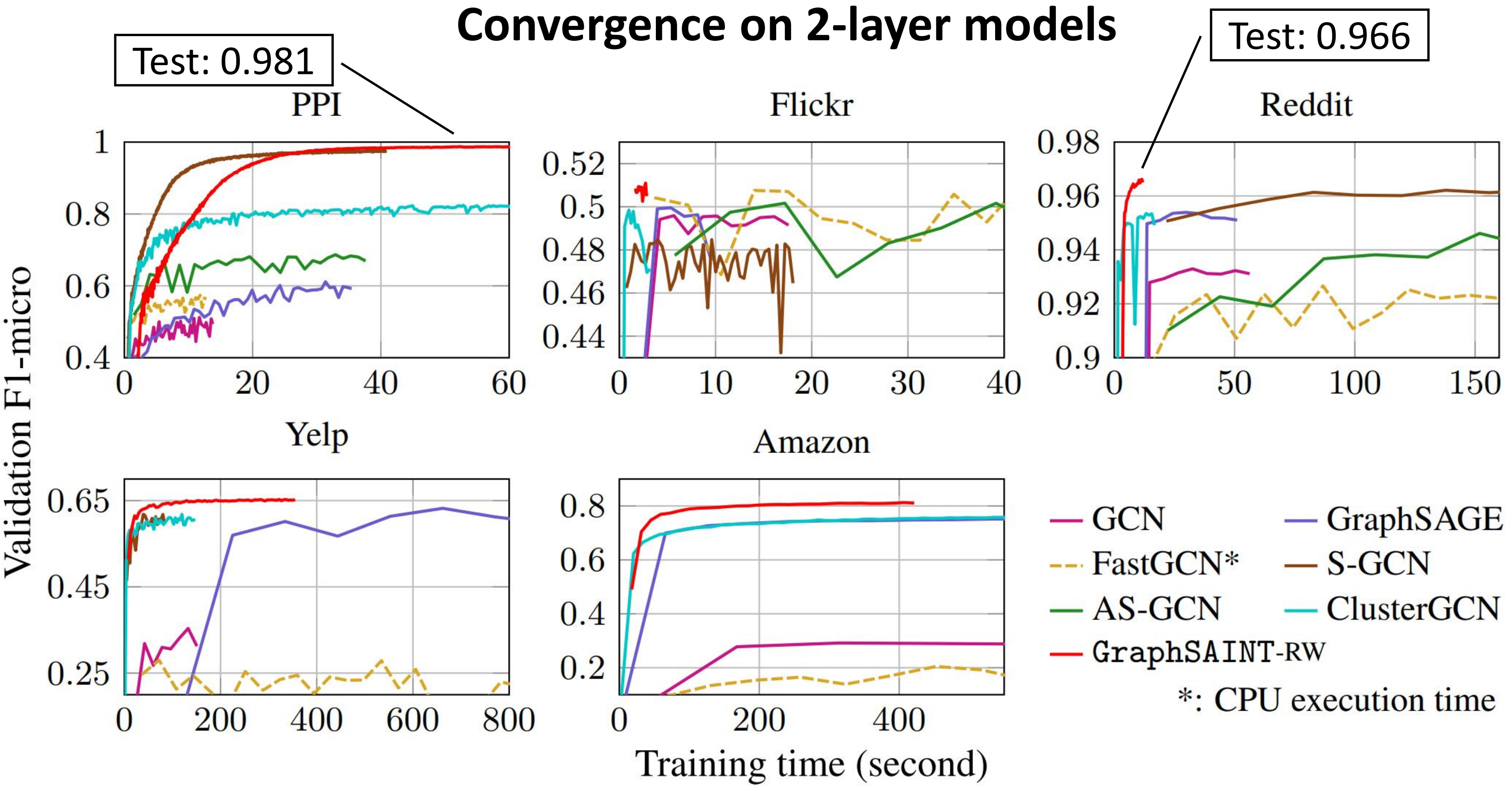
Experiments

Setup

- 5 large graphs — Up to 1.5M nodes
- 5 GNN architectures — GCN/GraphSAGE, GAT, JK-net, MixHop, GaAN etc.
- 4 graph samplers — Node, Edge, Random walk, MD-Random walk
- Multiple depths — From 2-layer to 5-layer GNNs

Dataset	Nodes	Edges	Degree	Feature	Classes	Train / Val / Test
PPI	14,755	225,270	15	50	121 (m)	0.66 / 0.12 / 0.22
Flickr	89,250	899,756	10	500	7 (s)	0.50 / 0.25 / 0.25
Reddit	232,965	11,606,919	50	602	41 (s)	0.66 / 0.10 / 0.24
Yelp	716,847	6,977,410	10	300	100 (m)	0.75 / 0.10 / 0.15
Amazon	1,598,960	132,169,734	83	200	107 (m)	0.85 / 0.05 / 0.10
PPI (large version)	56,944	818,716	14	50	121 (m)	0.79 / 0.11 / 0.10

Experiments

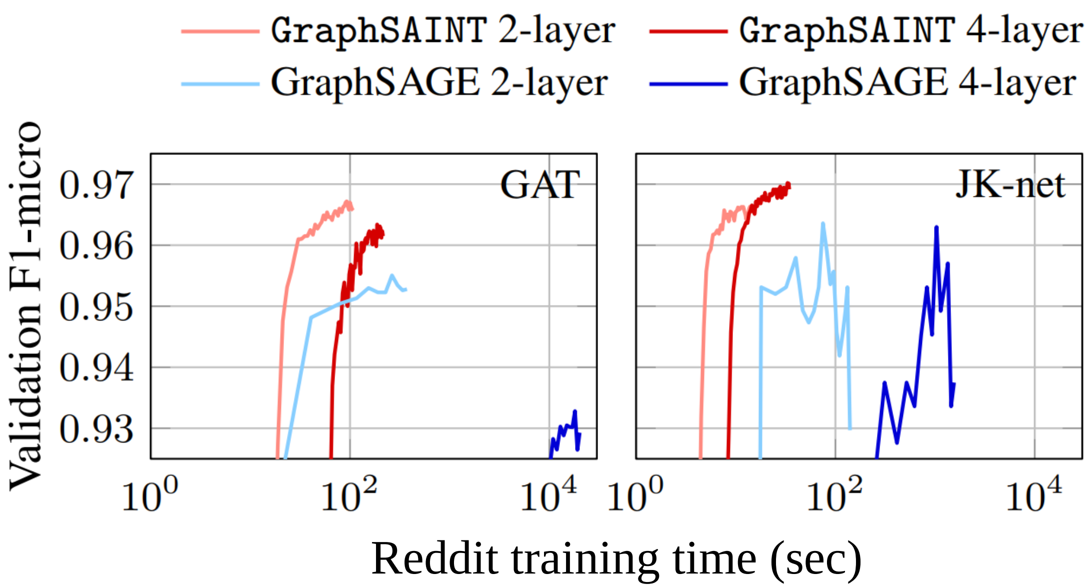


Experiments

Graph clustering *vs.* Graph sampling

	PPI (large version)		Reddit	
	2×512	5×2048	2×128	4×128
ClusterGCN	0.903±0.002	0.994±0.000	0.954±0.001	0.966±0.001
GraphSAINT	**0.941**±0.003	**0.995**±0.000	**0.966**±0.001	**0.970**±0.001

Layer sampling *vs.* Graph sampling



Conclusion

Minibatch training method for *deep* GNNs on *large* graphs

State-of-the-art training quality (accuracy, speed)

Highly flexible and extensible (graph samplers, GNN architectures)

Official code: https://github.com/GraphSAINT/GraphSAINT

GraphSAINT / **GraphSAINT** Watch 5 Star 63 Fork 13

PYTORCH TensorFlow

Reference implementation: **PyTorch Geometric** library

rusty1s / **pytorch_geometric** Watch 197 Star 7.3k Fork 1.2k

PYTORCH

References

Papers on slide 2

- **GraphSAGE**: "Inductive representation learning on large graphs". In NeurIPS 2017.
- **FastGCN**: "FastGCN: Fast learning with graph convolutional networks via importance sampling". In ICLR 2018.
- **S-GCN**: "Stochastic training of graph convolutional networks with variance reduction". In ICML 2018.
- **AS-GCN**: "Adaptive sampling towards fast graph representation learning". In NeurIPS 2019.
- **ParaGCN**: "Accurate, efficient and scalable graph embedding". In IEEE/IPDPS 2019.
- **Cluster-GCN**: "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks". In KDD 2019.
- **GraphSAINT**: "GraphSAINT: Graph sampling based inductive learning method". In ICLR 2020.

Thank you!