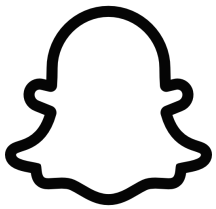
Scaling up Graph Neural Networks at Snap

Neil Shah

August 15, 2022

DLG-KDD 2022



Snap Inc.

Our Brands

Snapchat

Snapchat is a new kind of camera used by millions of people every day to stay in touch with friends, express themselves, explore the world — and take some pictures, too.

Spectacles

Spectacles are sunglasses that capture your world, the way you see it — and empower you to share your perspective with the world in a whole new way.

Bitmoji

Bitmoji is the digital you — a living cartoon character to instantly express who you are and how you're feeling, in the moment.

Why Snap?

319 million
daily active users (DAUs) use Snapchat every day on average.[1]

Over 6 billion
AR Lens plays per day on average.[1]

Over 200 million
DAUs engage with augmented reality every day on average.[1]

Over 250,000
Lens creators have used Lens Studio.[2]

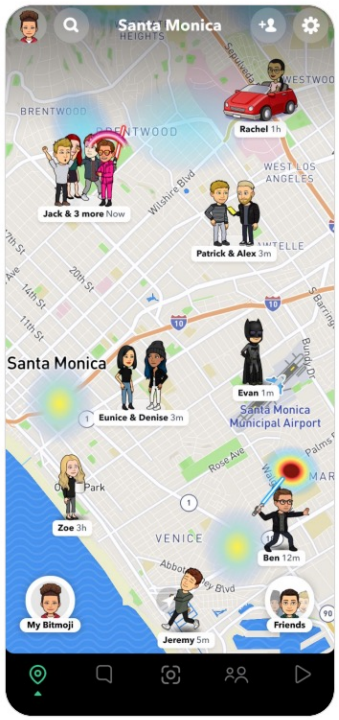
Over 2.5 million
Lenses made by our community.[2]

Over 75%
of the 13-34 year olds in the US, UK, Australia, France, and Netherlands use Snapchat.[1]

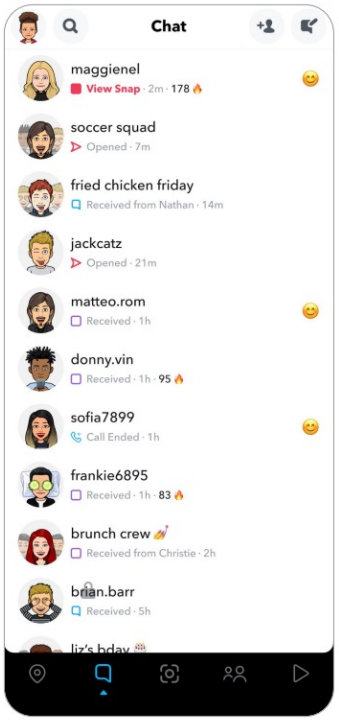
The Snapchat Experience

Five core platforms

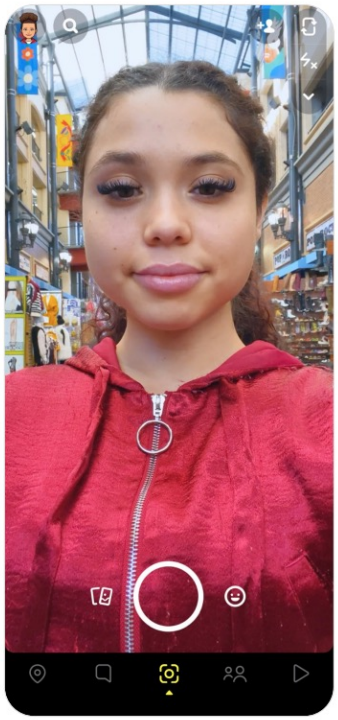
MAP



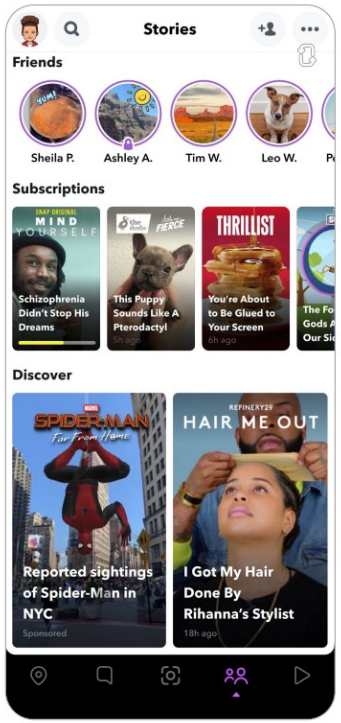
COMMUNICATIONS



CAMERA



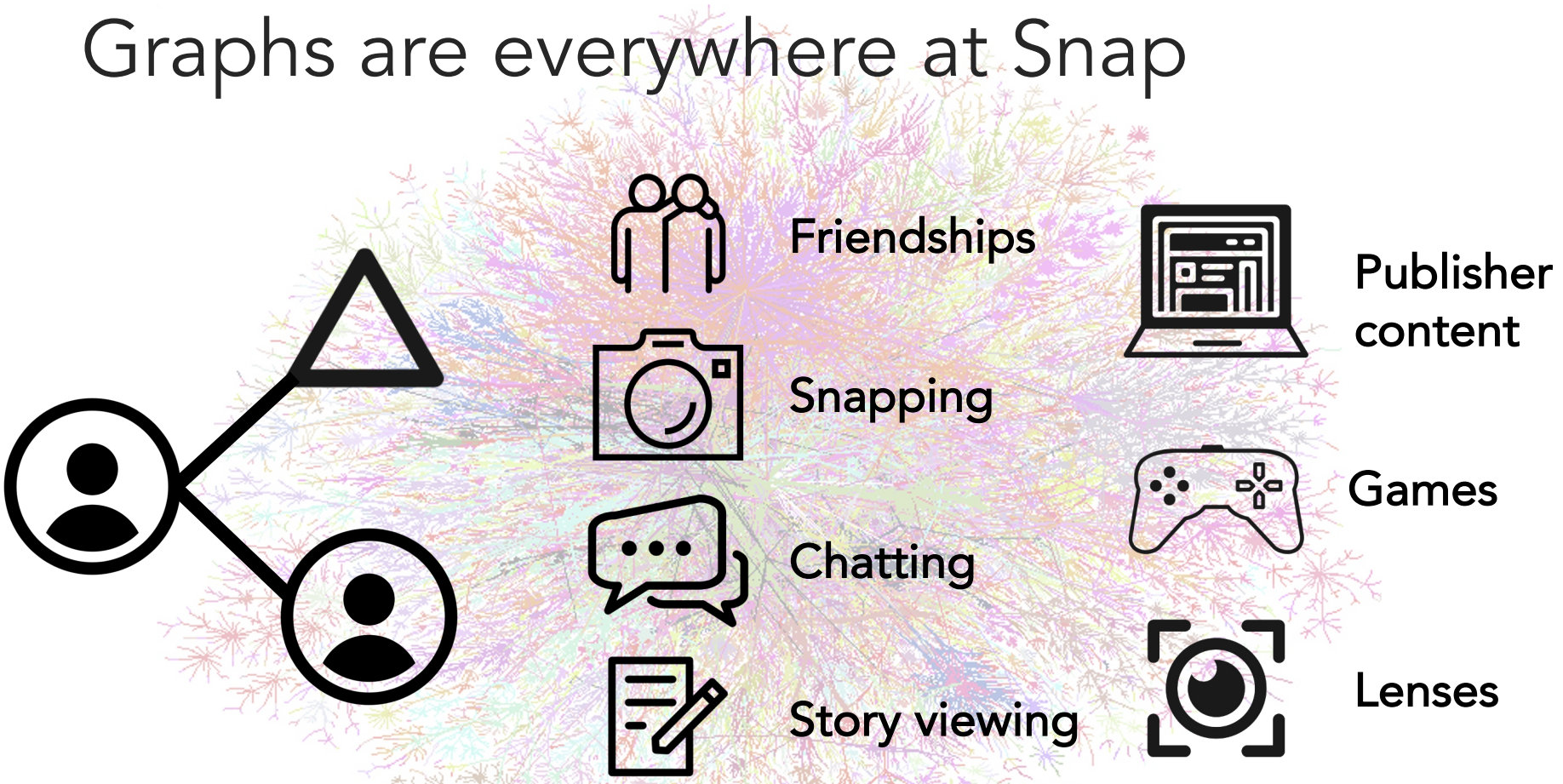
STORIES



SPOTLIGHT

February 2022 Investor Presentation (investor.snap.com)

Graphs are everywhere at Snap



Hundreds of millions of nodes, and billions of edges.

Graph Neural Networks

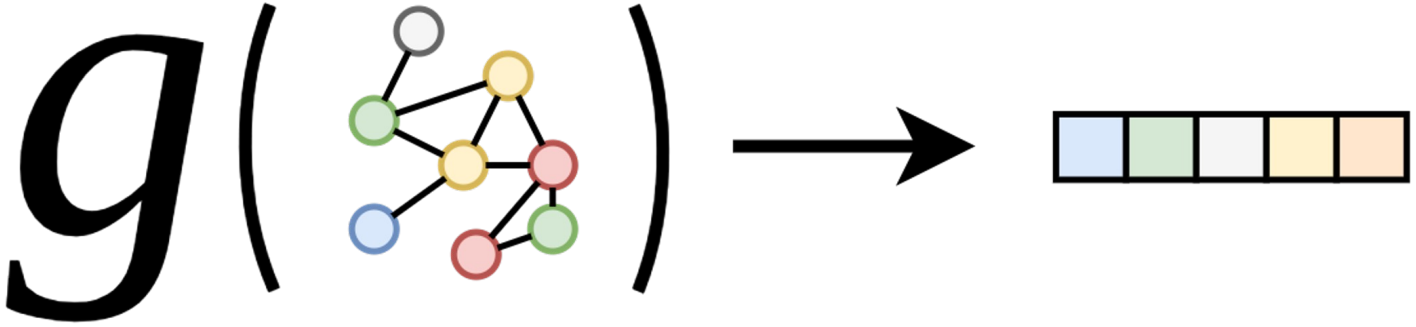


Figure credit: Lingxiao Zhao

Graph Neural Networks

- **Key idea:** stack message passing layers to learn node and / or graph embeddings

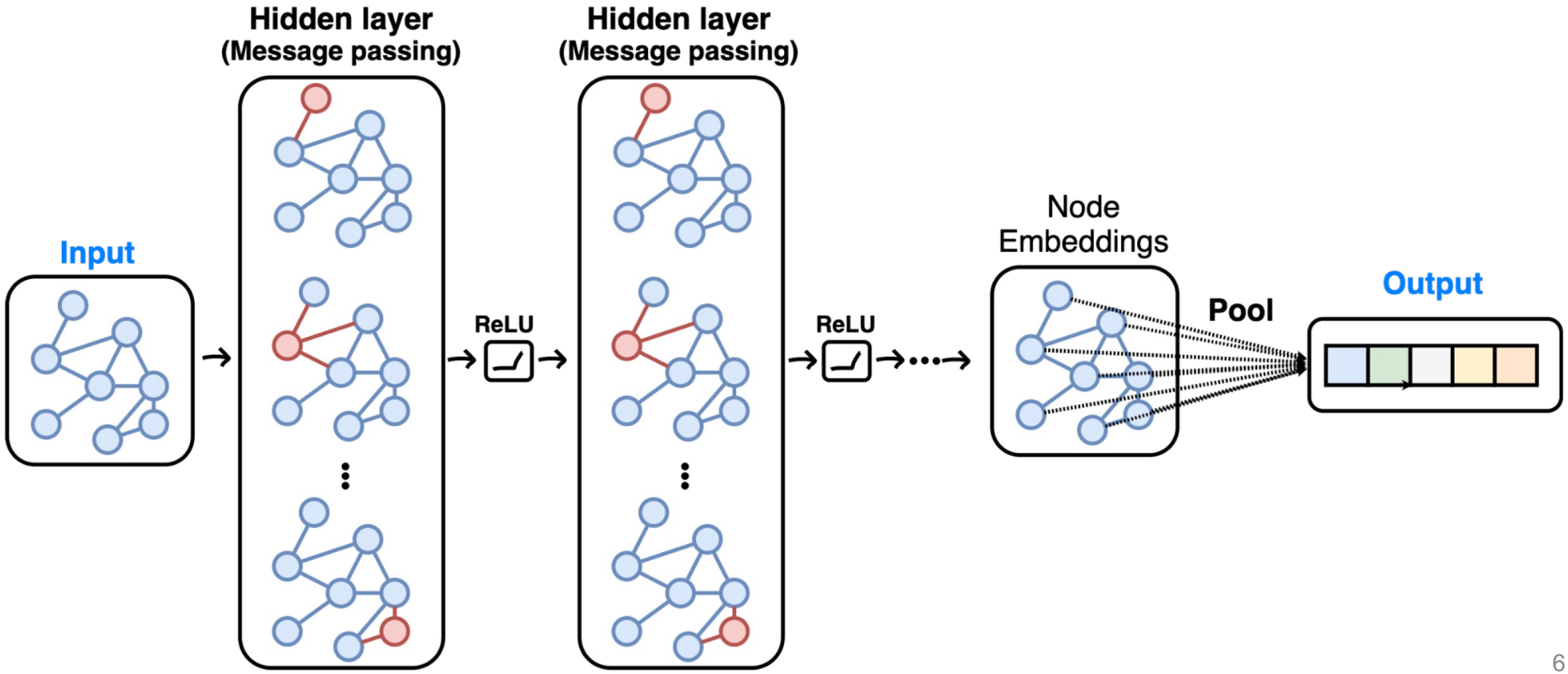


Figure credit: Lingxiao Zhao

Graph Neural Network (1 layer)

- (t-th) Message Passing Layer

$$h_v^{(t)} = \underbrace{\mathrm{AGG}^{(t)}}_{\text{Pooling}}\Big(\underbrace{h_v^{(t-1)}}_{\text{Node emb.}}, \underbrace{\Big\{\mathrm{MSG}^{(t)}(h_u^{(t-1)}) | u \in \mathcal{N}(v)\Big\}}_{\text{Neighbor embs.}}\Big)$$

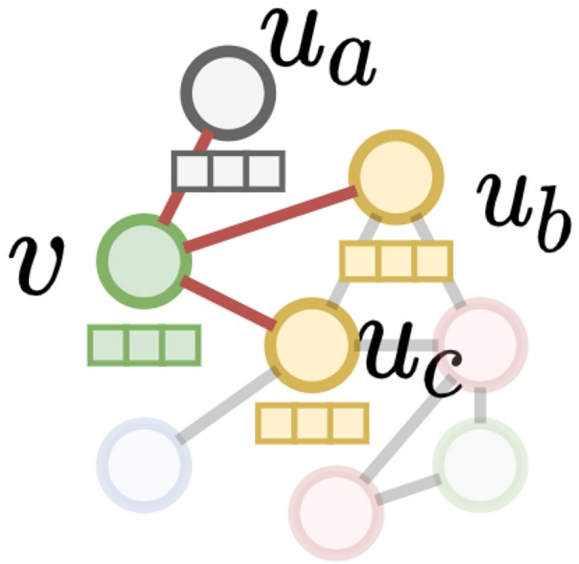
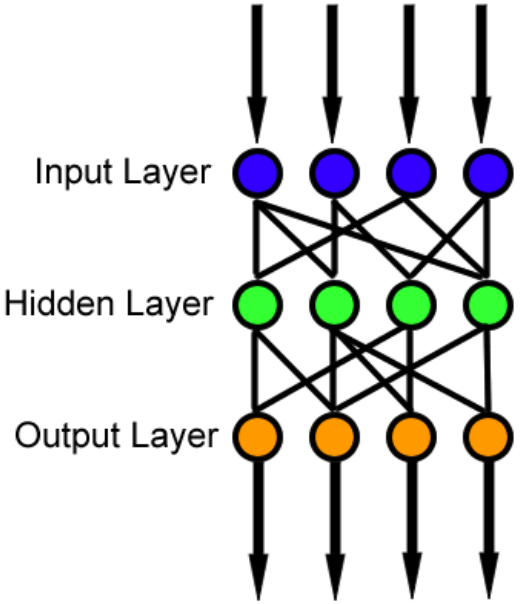
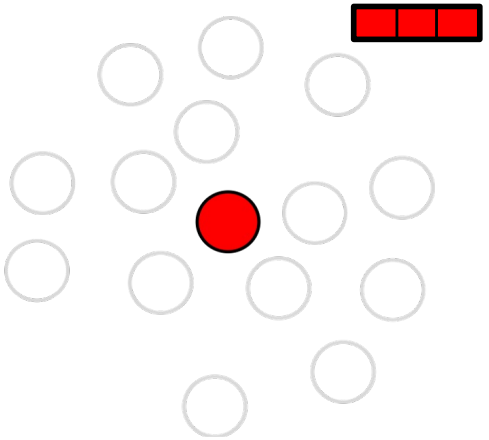


Figure credit: Lingxiao Zhao

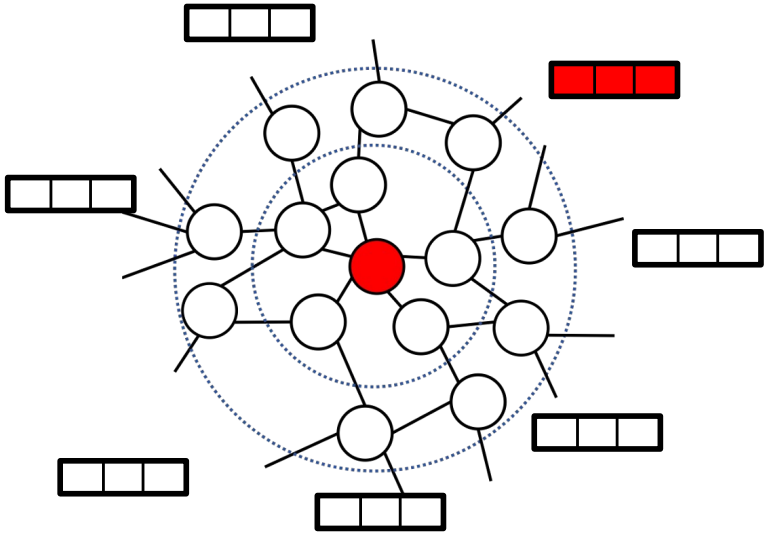
Relations to other approaches



Multi-layer perceptrons
(tabular machine learning)

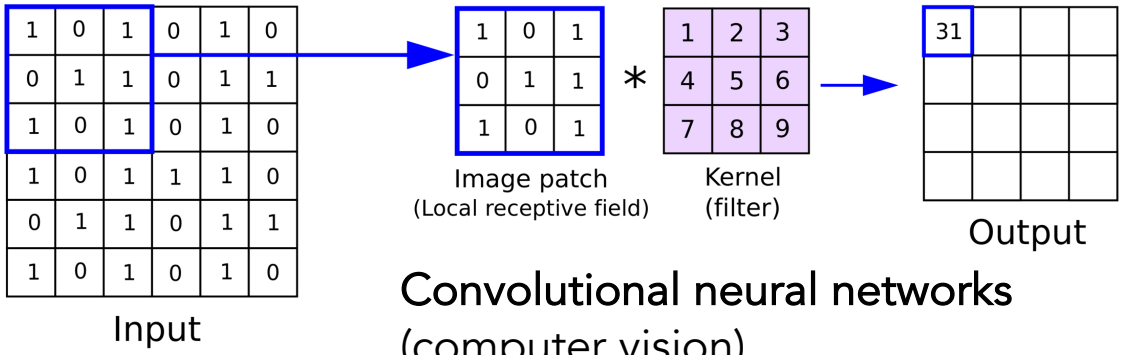


Tabular ML context

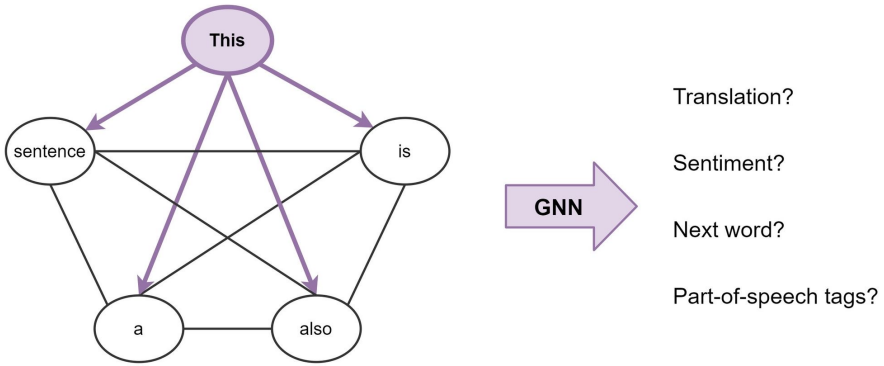


Graph ML context

Relations to other approaches



Convolutional neural networks
(computer vision)



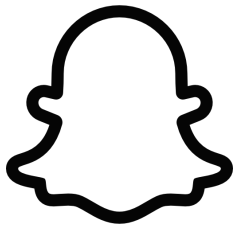
Transformers
(natural language processing)

Challenges in scaling GNNs

- Storing very large graphs is non-trivial
- Models are slow to train due to data dependency
- Realtime inference is slower than traditional models

…

Today: how can we scale up GNN training & inference?



Graph Condensation for GNNs

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, Neil Shah

ICLR 2022

Followup: Condensing Graphs via One-Step Gradient Descent (Jin et al, KDD'22)

Thursday, August 18, 10:00 AM-12:00 PM, Room 206, (Graph Mining)

Challenges in scaling GNNs

- Storing very large graphs is non-trivial
- Models are slow to train due to data dependency
- Realtime inference is slower than traditional models

...

Can we significantly speed up GNN training by making large graph data smaller?

Our problem: Graph Condensation

Given graph $\mathcal{T} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$, we aim to learn a graph $\mathcal{S} = (\mathbf{A}', \mathbf{X}', \mathbf{Y}')$ with fewer nodes such that a GNN trained on $\mathcal{S}$ can achieve comparable performance to one trained on $\mathcal{T}$.

$$\min_{\mathcal{S}} \mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}_{\mathcal{S}}}(\mathbf{A}, \mathbf{X}), \mathbf{Y}\right)$$

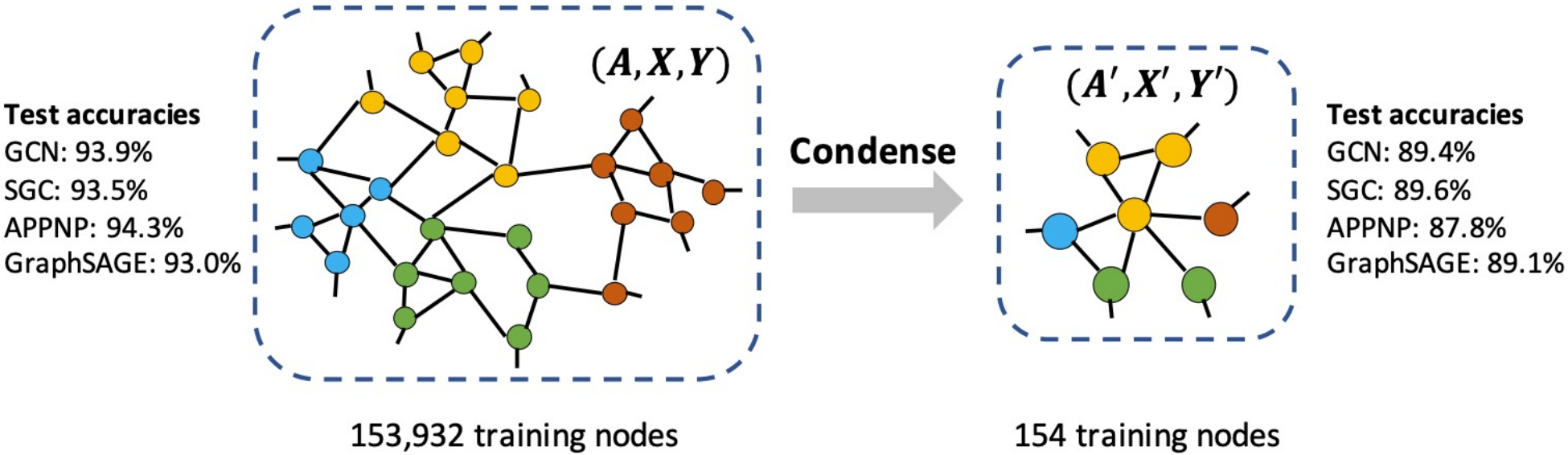
Minimize loss on original graph

$$\text{s.t} \quad \boldsymbol{\theta}_{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathrm{GNN}_{\boldsymbol{\theta}}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}')$$

via learned parameters on condensed graph

Our problem: Graph Condensation

Given graph $\mathcal{T} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$, we aim to learn a graph $\mathcal{S} = (\mathbf{A}', \mathbf{X}', \mathbf{Y}')$ with fewer nodes such that a GNN trained on $\mathcal{S}$ can achieve comparable performance to one trained on $\mathcal{T}$.



Proposal: Graph Condensation via Gradient Matching

- Directly optimizing the bi-level problem is hard!
- Instead, try to learn $\mathcal{S}$, such that $\theta^{\mathcal{S}} \approx \theta^{\mathcal{T}}$

$$\min_{\mathcal{S}} \left[\sum_{t=0}^{T-1} D\left(\boldsymbol{\theta}_t^{\mathcal{S}}, \boldsymbol{\theta}_t^{\mathcal{T}}\right) \right] \quad \text{with}$$

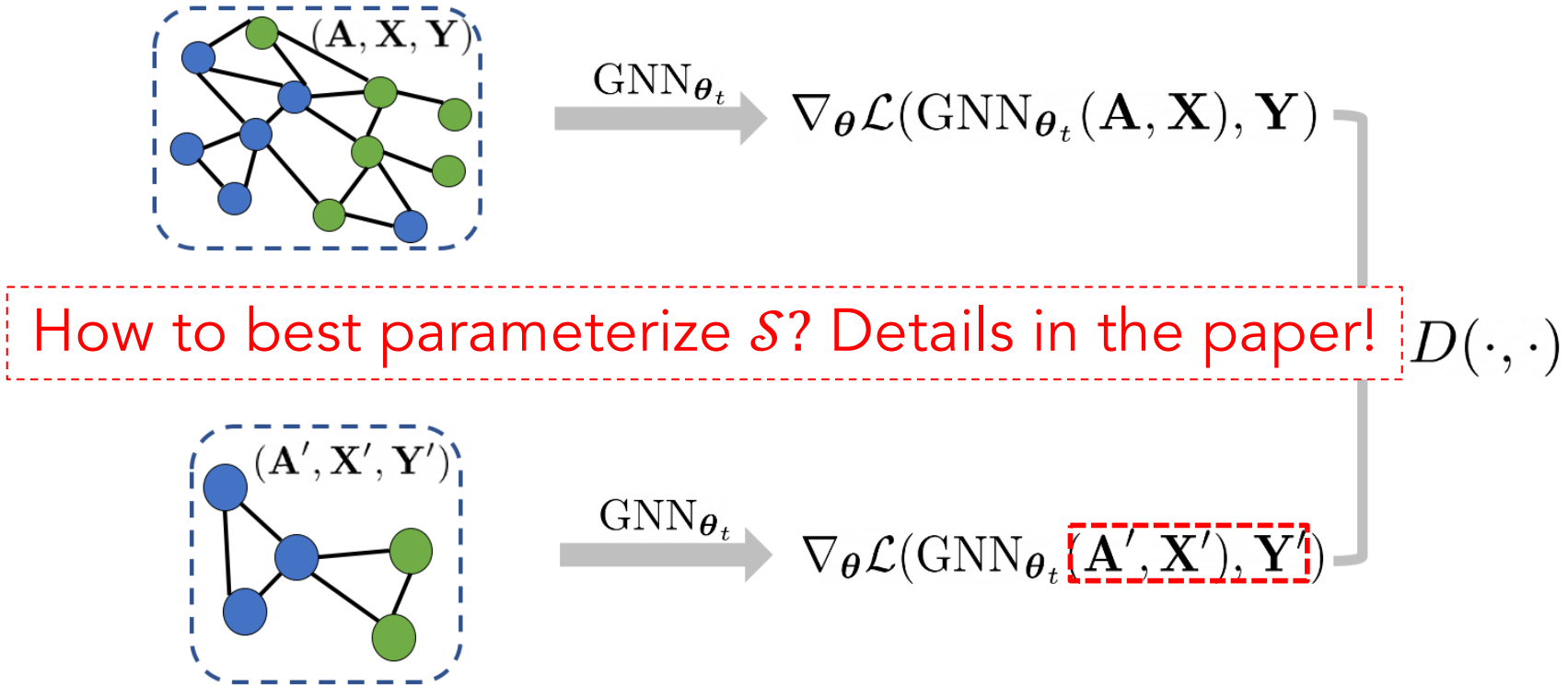
$$\underbrace{\boldsymbol{\theta}_{t+1}^{\mathcal{S}} = \text{opt}_{\boldsymbol{\theta}}\left(\mathcal{L}\left(\text{GNN}_{\boldsymbol{\theta}_t^{\mathcal{S}}}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'\right)\right)}_{\text{Params from condensed graph}} \quad \text{and} \quad \underbrace{\boldsymbol{\theta}_{t+1}^{\mathcal{T}} = \text{opt}_{\boldsymbol{\theta}}\left(\mathcal{L}\left(\text{GNN}_{\boldsymbol{\theta}_t^{\mathcal{T}}}(\mathbf{A}, \mathbf{X}), \mathbf{Y}\right)\right)}_{\text{Params from original graph}}$$

- We convert the problem to matching gradients[1]:

$$\min_{\mathcal{S}} \left[\sum_{t=0}^{T-1} D\left(\underbrace{\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\text{GNN}_{\boldsymbol{\theta}_t}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'\right)}_{\text{Gradients from condensed graph}}, \underbrace{\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\text{GNN}_{\boldsymbol{\theta}_t}(\mathbf{A}, \mathbf{X}), \mathbf{Y}\right)}_{\text{Gradients from original graph}}\right) \right]$$

[1] Dataset Condensation with Gradient Matching (Zhao et al, 2021)

Proposal: Graph Condensation via Gradient Matching

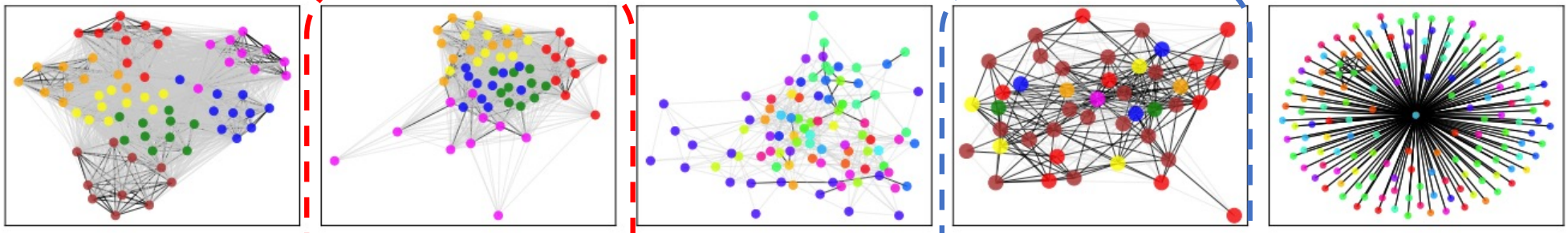


Training with condensed graphs is comparable to training on their original counterparts.

Dataset	Ratio (r)	Baselines					Proposed		Whole Dataset
		Random $(\mathbf{A}', \mathbf{X}')$	Herding $(\mathbf{A}', \mathbf{X}')$	K-Center $(\mathbf{A}', \mathbf{X}')$	Coarsening $(\mathbf{A}', \mathbf{X}')$	DC-Graph $(\mathbf{X}')$	GCOND-X $(\mathbf{X}')$	GCOND $(\mathbf{A}', \mathbf{X}')$	
Citeseer	0.9%	54.4±4.4	57.1±1.5	52.4±2.8	52.2±0.4	66.8±1.5	**71.4**±0.8	70.5±1.2	71.7±0.1
	1.8%	64.2±1.7	66.7±1.0	64.3±1.0	59.0±0.5	66.9±0.9	69.8±1.1	**70.6**±0.9	
	3.6%	69.1±0.1	69.0±0.1	69.1±0.1	65.3±0.5	66.3±1.5	69.4±1.4	**69.8**±1.4	
Cora	1.3%	63.6±3.7	67.0±1.3	64.0±2.3	31.2±0.2	67.3±1.9	75.9±1.2	**79.8**±1.3	81.2±0.2
	2.6%	72.8±1.1	73.4±1.0	73.2±1.2	65.2±0.6	67.6±3.5	75.7±0.9	**80.1**±0.6	
	5.2%	76.8±0.1	76.8±0.1	76.7±0.1	70.6±0.1	67.7±2.2	76.0±0.9	**79.3**±0.3	
Ogbn-arxiv	0.05%	47.1±3.9	52.4±1.8	47.2±3.0	35.4±0.3	58.6±0.4	**61.3**±0.5	59.2±1.1	71.4±0.1
	0.25%	57.3±1.1	58.6±1.2	56.8±0.8	43.5±0.2	59.9±0.3	**64.2**±0.4	63.2±0.3	
	0.5%	60.0±0.9	60.4±0.8	60.3±0.4	50.4±0.1	59.5±0.3	63.1±0.5	**64.0**±0.4	
Flickr	0.1%	41.8±2.0	42.5±1.8	42.0±0.7	41.9±0.2	46.3±0.2	45.9±0.1	**46.5**±0.4	47.2±0.1
	0.5%	44.0±0.4	43.9±0.9	43.2±0.1	44.5±0.1	45.9±0.1	45.0±0.2	**47.1**±0.1	
	1%	44.6±0.2	44.4±0.6	44.1±0.4	44.6±0.1	45.8±0.1	45.0±0.1	**47.1**±0.1	
Reddit	0.05%	46.1±4.4	53.1±2.5	46.6±2.3	40.9±0.5	88.2±0.2	**88.4**±0.4	88.0±1.8	93.9±0.0
	0.1%	58.0±2.2	62.7±1.0	53.0±3.3	42.8±0.8	89.5±0.1	89.3±0.1	**89.6**±0.7	
	0.2%	66.3±1.9	71.0±1.6	58.5±2.1	47.4±0.9	**90.5**±1.2	88.8±0.4	90.1±0.5	

Condensed graphs are concise and can capture original graph properties.

	Citeseer, r=0.9%		Cora, r=1.3%		Ogbn-arxiv, r=0.25%		Flickr, r=0.5%		Reddit, r=0.1%	
	Whole	GCOND	Whole	GCOND	Whole	GCOND	Whole	GCOND	Whole	GCOND
Accuracy	70.7	70.5	81.5	79.8	71.4	63.2	47.1	47.1	94.1	89.4
Homophily	0.74	0.65	0.81	0.79	0.65	0.07	0.33	0.28	0.78	0.04
Storage	47.1 MB	0.9 MB	14.9 MB	0.4 MB	100.4 MB	0.3 MB	86.8 MB	0.5 MB	435.5 MB	0.4 MB



(a) Cora, r=2.5% (b) Citeseer, r=1.8% (c) Arxiv, r=0.05% (d) Flickr, r=0.1% (e) Reddit, r=0.1%

Condensed graphs can be used to well-train different GNN architectures.

	Methods	Data	MLP	GAT	APPNP	Cheby	GCN	SAGE	SGC	Avg.	Whole Dataset
Citeseer $r = 1.8\%$	DC-Graph	$\mathbf{X}'$	66.2	-	66.4	64.9	66.2	65.9	69.6	66.6	71.7±0.1
	GCOND-X	$\mathbf{X}'$	69.6	-	69.7	70.6	69.7	69.2	71.6	70.2	
	GCOND	$\mathbf{A}', \mathbf{X}'$	63.9	55.4	69.6	68.3	70.5	66.2	70.3	69.0	
Cora $r = 2.6\%$	DC-Graph	$\mathbf{X}'$	67.2	-	67.1	67.7	67.9	66.2	72.8	68.3	81.2±0.2
	GCOND-X	$\mathbf{X}'$	76.0	-	77.0	74.1	75.3	76.0	76.1	75.7	
	GCOND	$\mathbf{A}', \mathbf{X}'$	73.1	66.2	78.5	76.0	80.1	78.2	79.3	78.4	
Ogbn-arxiv $r = 0.25\%$	DC-Graph	$\mathbf{X}'$	59.9	-	60.0	55.7	59.8	60.0	60.4	59.2	71.4±0.1
	GCOND-X	$\mathbf{X}'$	64.1	-	61.5	59.5	64.2	64.4	64.7	62.9	
	GCOND	$\mathbf{A}', \mathbf{X}'$	62.2	60.0	63.4	54.9	63.2	62.6	63.7	61.6	
Flickr $r = 0.5\%$	DC-Graph	$\mathbf{X}'$	43.1	-	45.7	43.8	45.9	45.8	45.6	45.4	47.2±0.1
	GCOND-X	$\mathbf{X}'$	42.1	-	44.6	42.3	45.0	44.7	44.4	44.2	
	GCOND	$\mathbf{A}', \mathbf{X}'$	44.8	40.1	45.9	42.8	47.1	46.2	46.1	45.6	
Reddit $r = 0.1\%$	DC-Graph	$\mathbf{X}'$	50.3	-	81.2	77.5	89.5	89.7	90.5	85.7	93.9±0.0
	GCOND-X	$\mathbf{X}'$	40.1	-	78.7	74.0	89.3	89.3	91.0	84.5	
	GCOND	$\mathbf{A}', \mathbf{X}'$	42.5	60.2	87.8	75.5	89.4	89.1	89.6	86.3	

GCN performance

[1] A Unified View of Graph Neural Networks on Graph Signal Denoising (Ma et al, 2021)

Condensed graphs can be used for Neural Architecture Search.

	Random	Herding	GCOND
Cora	0.40	0.21	**0.76**
Citeseer	0.56	0.29	**0.79**
Ogbn-arxiv	0.63	0.6	**0.64**

Pearson correlation between condensed graph and original graph validation accuracies of 480 GNN models.

Graph-less Neural Networks

Shichang Zhang, Yozen Liu, Yizhou Sun, Neil Shah

ICLR 2022

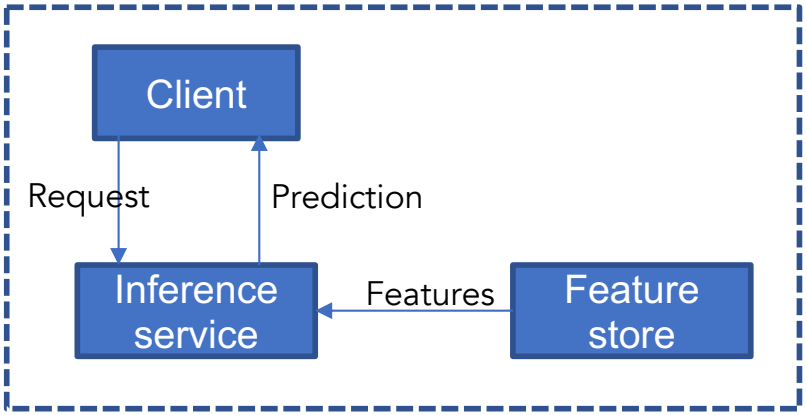
Challenges in scaling GNNs

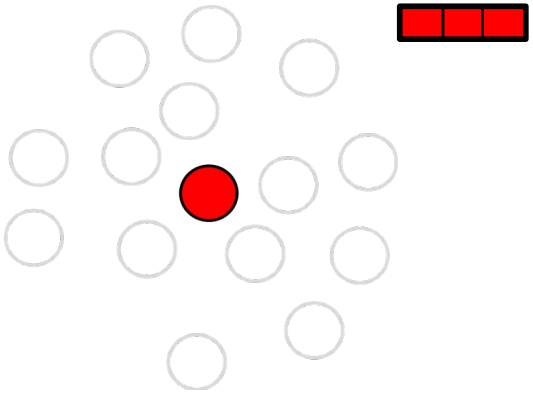
- Storing very large graphs is non-trivial
- Models are slow to train due to data dependency
- Realtime inference is slower than traditional models

...

Can we deploy GNNs in real-time inference settings without latency overheads?

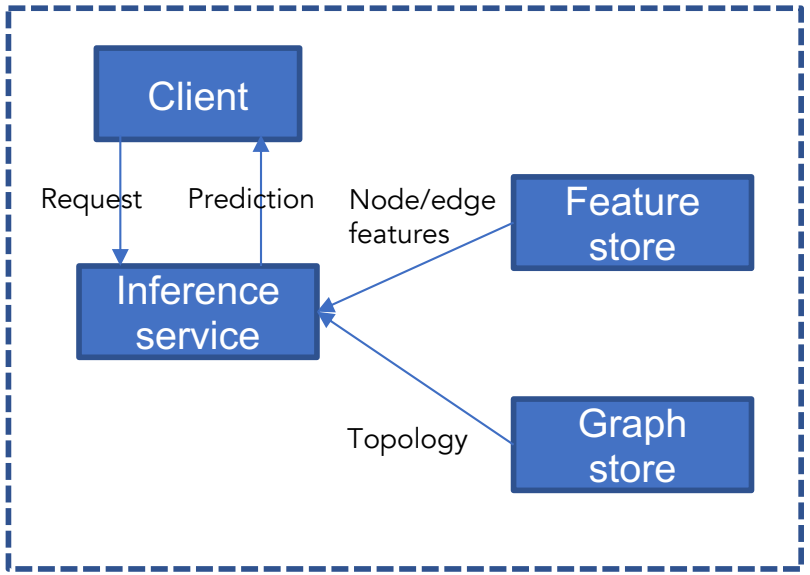
Context: (Tabular) Real-time inference

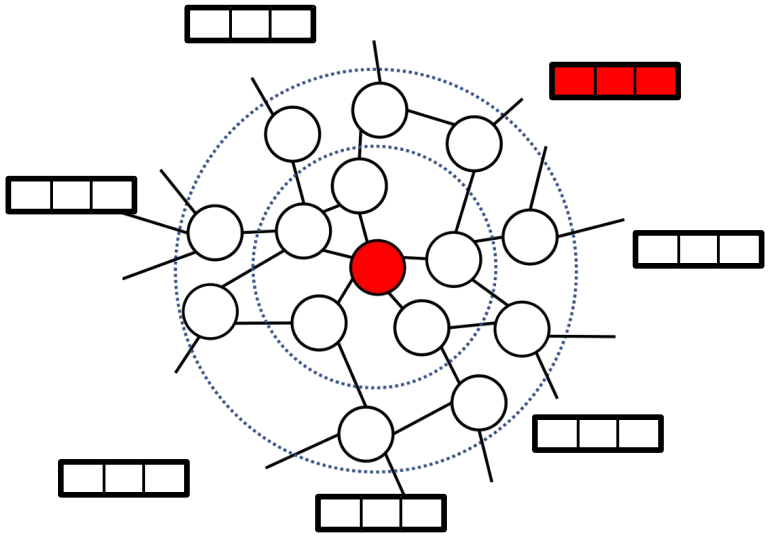




- Latency-constrained applications
- Feature data too large for a single machine
- MLPs are a workhorse of most such modern systems

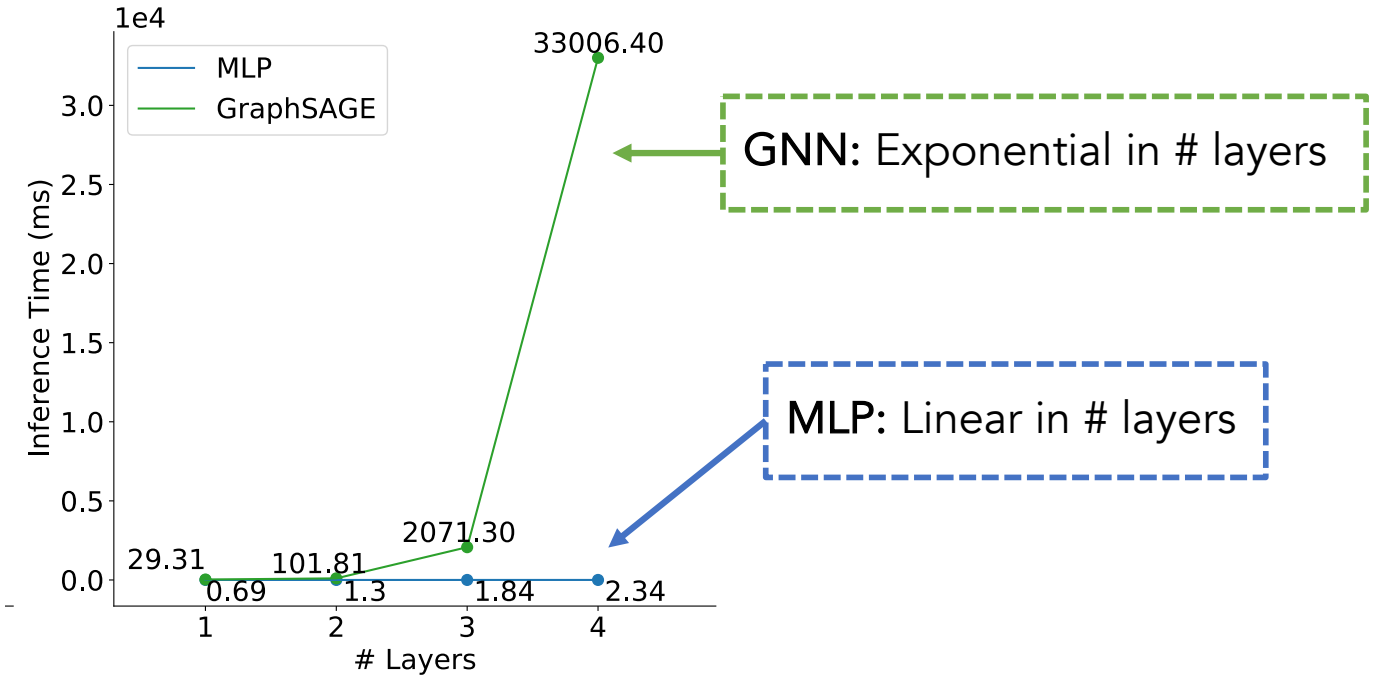
Context: (Graph) Real-time inference





- Large graphs, with high average-degree, in practice
- Requires **many** queries to facilitate message passing

GNNs are way slower than MLPs…



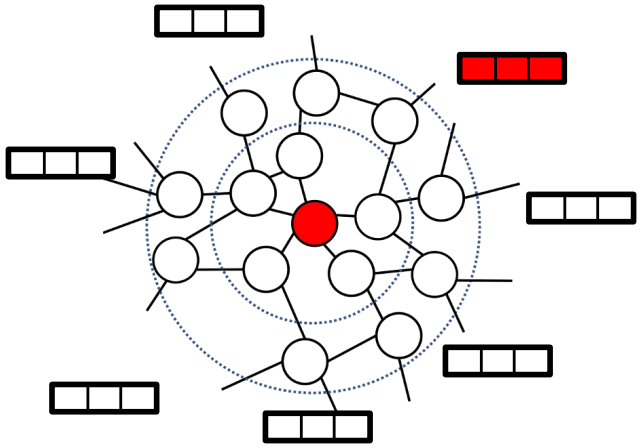
- Infer 10 randomly selected nodes (graph of 2.5M nodes)
- Inference time = **fetching data** + forward pass

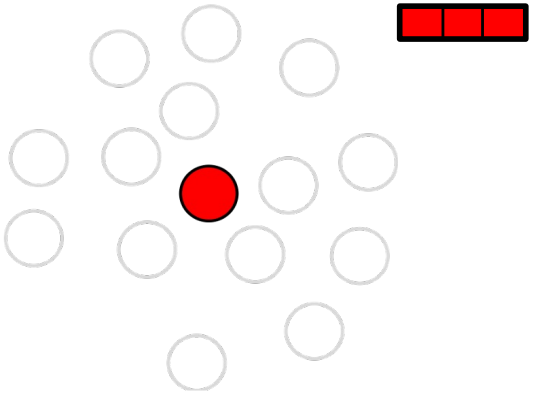
…but also way more accurate!

Datasets	GraphSAGE	MLP
Cora	80.52 ± 1.77	59.22 ± 1.31
Citeseer	70.33 ± 1.97	59.61 ± 2.88
Pubmed	75.39 ± 2.09	67.55 ± 2.31
A-computer	82.97 ± 2.16	67.80 ± 1.06
A-photo	90.90 ± 0.84	78.77 ± 1.74
Arxiv	70.92 ± 0.17	56.05 ± 0.46
Products	78.61 ± 0.49	62.47 ± 0.10

Node classification accuracy on seven benchmarks.

GNN and MLP: Combining Advantages





Accurate GNN:

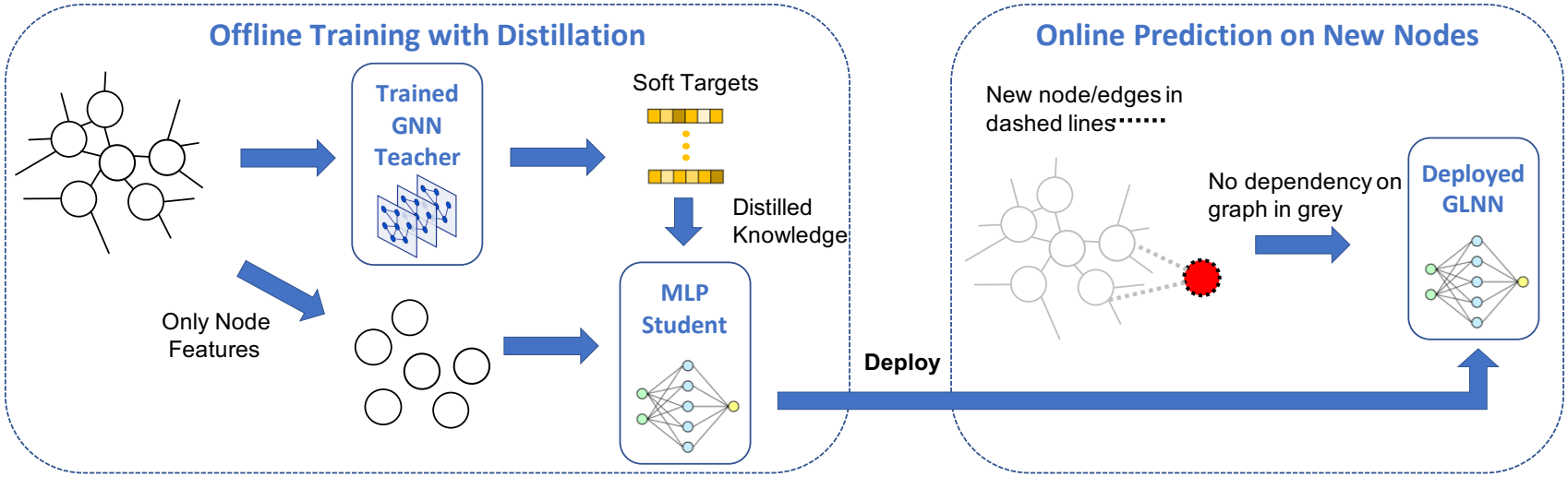
- Graph dependence in training
- Graph dependence in inference

Fast MLP:

- No graph dependence in training
- No graph dependence in inference

Can we use graph dependency in training, but not inference?

Proposal: Graph-less Neural Networks (GLNNs)



- **Offline training:** *use graph* to train GNN; distill to MLP
- **Online prediction:** faster, more accurate *graph-less* inference for new nodes

Proposal: Graph-less Neural Networks (GLNNs)

1. Train a (teacher) GNN, and produce logits for each node z_v.

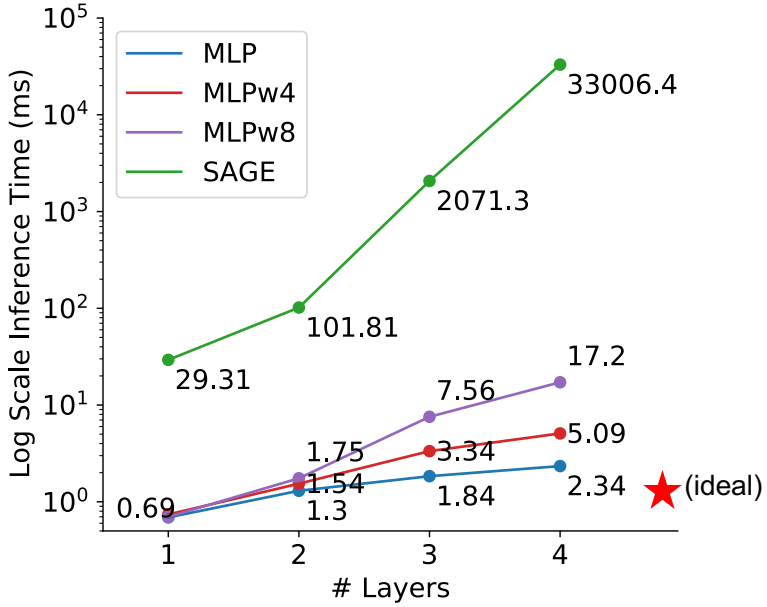
2. Train a (student) GLNN using the below loss.

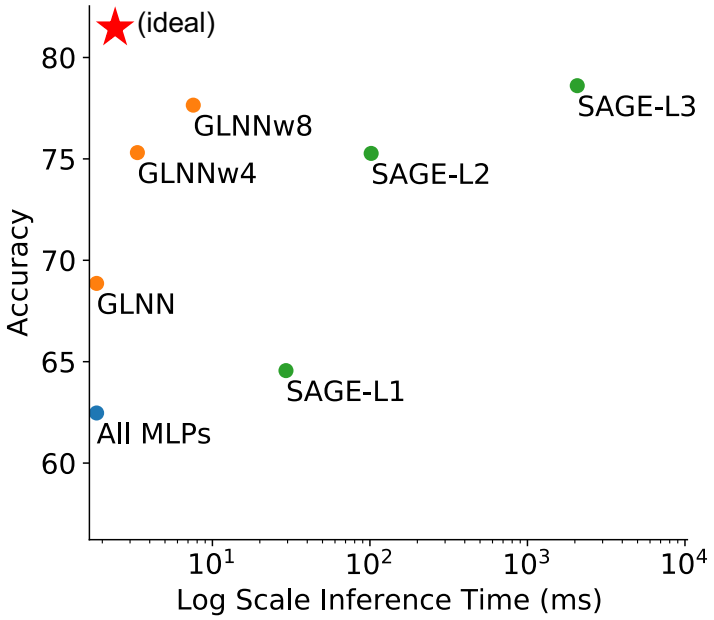
$$\mathcal{L} = \lambda \Sigma_{v \in \mathcal{V}^L} \mathcal{L}_{label}(\hat{\boldsymbol{y}}_v, \boldsymbol{y}_v) + (1-\lambda) \Sigma_{v \in \mathcal{V}} \mathcal{L}_{teacher}(\hat{\boldsymbol{y}}_v, \boldsymbol{z}_v)$$

Encourage MLP to correctly predict labeled nodes from logits

Encourage MLP/GNN logit consistency for all nodes

Trade-offs Between Speed and Accuracy





- MLPs/GLNNs are **much faster** than GNNs
- This holds even with wider MLPs/GLNNs

- GLNN is **much more accurate** than MLP
- GLNNs are **comparably accurate** to GNNs while being **much faster**

GLNNs empower MLPs to be like GNNs.

Datasets	Eval	SAGE	MLP/MLP+	GLNN/GLNN+	Δ_{MLP}	Δ_{GNN}
Cora	*prod*	**79.29**	58.98	78.28	19.30 (32.72%)	-1.01 (-1.28%)
	ind	81.33 ± 2.19	59.09 ± 2.96	73.82 ± 1.93	14.73 (24.93%)	-7.51 (-9.23%)
	tran	78.78 ± 1.92	58.95 ± 1.66	79.39 ± 1.64	20.44 (34.66%)	0.61 (0.77%)
Citeseer	*prod*	68.38	59.81	**69.27**	9.46 (15.82%)	0.89 (1.30%)
	ind	69.75 ± 3.59	60.06 ± 5.00	69.25 ± 2.25	9.19 (15.30%)	-0.5 (-0.7%)
	tran	68.04 ± 3.34	59.75 ± 2.48	69.28 ± 3.12	9.63 (15.93%)	1.24 (1.82%)
Pubmed	*prod*	**74.88**	66.80	74.71	7.91 (11.83%)	-0.17 (-0.22%)
	ind	75.26 ± 2.57	66.85 ± 2.96	74.30 ± 2.61	7.45 (11.83%)	-0.96 (-1.27%)
	tran	74.78 ± 2.22	66.79 ± 2.90	74.81 ± 2.39	8.02 (12.01%)	0.03 (0.04%)
A-computer	*prod*	82.14	67.38	**82.29**	14.90 (22.12%)	0.15 (0.19%)
	ind	82.08 ± 1.79	67.84 ± 1.78	80.92 ± 1.36	13.08 (19.28%)	-1.16 (-1.41%)
	tran	82.15 ± 1.55	67.27 ± 1.36	82.63 ± 1.40	15.36 (22.79%)	0.48 (0.58%)
A-photo	*prod*	91.08	79.25	**92.38**	13.13 (16.57%)	1.30 (1.42%)
	ind	91.50 ± 0.79	79.44 ± 1.72	91.18 ± 0.81	11.74 (14.78%)	-0.32 (-0.35%)
	tran	90.80 ± 0.77	79.20 ± 1.64	92.68 ± 0.56	13.48 (17.01%)	1.70 (1.87%)
Arxiv	*prod*	**70.73**	55.30	65.09	9.79 (17.70%)	-5.64 (-7.97%)
	ind	70.64 ± 0.67	55.40 ± 0.56	60.48 ± 0.46	4.3 (7.76%)	-10.94 (-15.49%)
	tran	70.75 ± 0.27	55.28 ± 0.49	71.46 ± 0.33	11.16 (20.18%)	-4.31 (-6.09%)
Products	*prod*	**76.60**	63.72	75.77	12.05 (18.91%)	-0.83 (-1.09%)
	ind	76.89 ± 0.53	63.70 ± 0.66	75.16 ± 0.34	11.44 (17.96%)	-1.73 (-2.25%)
	tran	76.53 ±0.55	63.73 ± 0.69	75.92 ± 0.61	12.20 (19.15%)	-0.61 (-0.79%)

Significant accuracy improvement over MLPs.

Competitive accuracy to GNNs on 6/7 datasets.

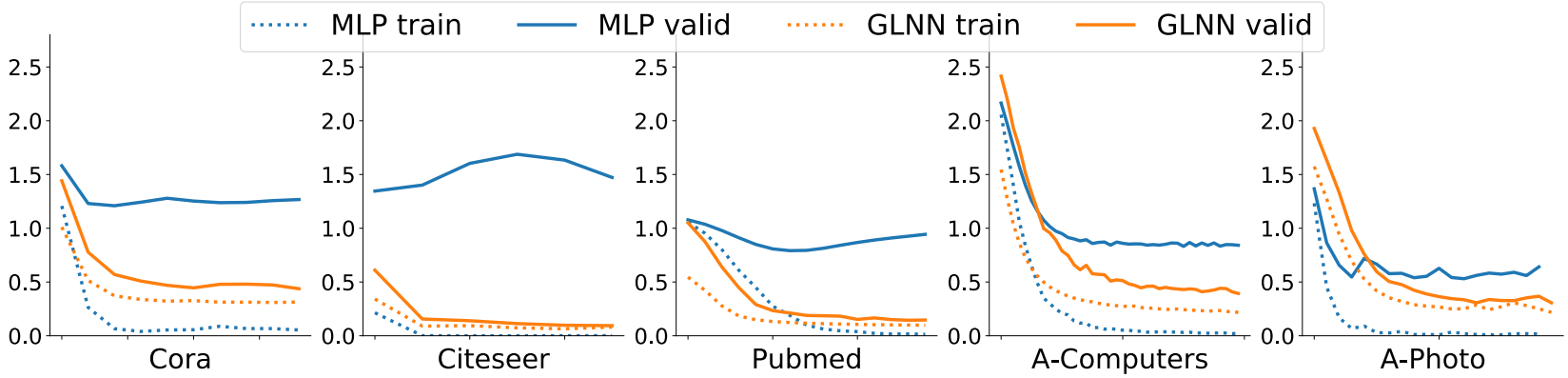
GLNNs are much faster than other inference acceleration methods.

Datasets	SAGE	QSAGE	PSAGE	Neighbor Sample	GLNN+
Arxiv	489.49	433.90 (1.13×)	465.43 (1.05×)	91.03 (5.37×)	**3.34 (146.55×)**
Products	2071.30	1946.49 (1.06×)	2001.46 (1.04×)	107.71 (19.23×)	**7.56 (273.98×)**

Inference time for 10 randomly chosen nodes.

- **SAGE:** base GNN model
- **QSAGE:** Quantized SAGE, FP32 to INT8
- **PSAGE:** Pruned SAGE, with 50% model parameters pruned
- **Neighbor Sampling:** sampling 15 nodes per layer

GNN KD helps regularize MLP training.



KD helps MLPs **match inductive bias** of GNNs.

$$\mathcal{L}_{cut} = \frac{Tr(\hat{\boldsymbol{Y}}^T \boldsymbol{A} \hat{\boldsymbol{Y}})}{Tr(\hat{\boldsymbol{Y}}^T \boldsymbol{D} \hat{\boldsymbol{Y}})} \qquad \mathcal{L}_{cut} \in [0, 1]$$

measures consistency between model prediction ($\hat{\boldsymbol{Y}}$) and graph topology ($\boldsymbol{A}$: adjacency matrix, $\boldsymbol{D}$: degree matrix)

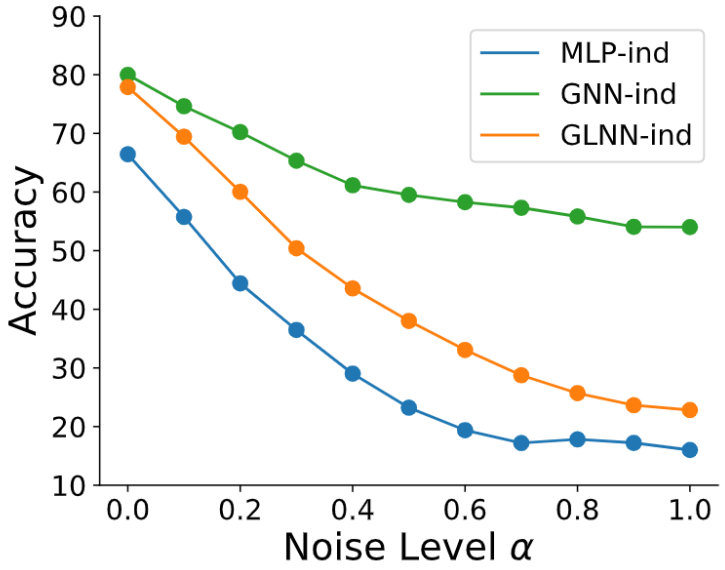
Datasets	SAGE	MLP	GLNN
Cora	0.9347	0.7026	0.8852
Citeseer	0.9485	0.7693	0.9339
Pubmed	0.9605	0.9455	0.9701
A-computer	0.9003	0.6976	0.8638
A-photo	0.8664	0.7069	0.8398
Average	0.9221	0.7644	0.8986

GLNNs aren't perfect.

GLNNs suffer when labels are less-correlated to node features (e.g. more correlated to structural aspects).

Add Gaussian noise to node features

$$\tilde{\boldsymbol{X}} = (1-\alpha)\boldsymbol{X} + \alpha\epsilon$$



As the correlation between labels and node features decreases…

- GNN degrades more gracefully by using graph structure information
- GLNN gets less accurate.

NB: In practical tasks, the node features and structural roles are often highly correlated.

Concluding remarks

Takeaways

- Scalable large-scale graph learning with GNNs is hard!
- Our recent work in condensation and distillation pushes boundaries in scalable training and inference.
- The work is never done; **Snap is hiring**!
- E-mail me at nshah@snap.com if you want to work together!

Snap Inc.