GIST: Distributed Training for Large-Scale Graph Convolutional Networks

Cameron R. Wolfe*[1], Jingkang Yang*[2], Arindam Chowdhury[3], Chen Dun[1], Artun Bayer[3], Santiago Segarra[3], and Anastasios Kyrillidis[1]

[1]Department of Computer Science, Rice University, Houston, TX, USA.
[2]School of Computer Science and Engineering, Nanyang Technology University, Singapore.
[3]Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA.

Main Idea: *We propose graph independent subnetwork training (GIST), a model-agnostic distributed training framework, to pioneer the training of ultra-wide graph convolutional network (GCN) models and reduce wall-clock time of GCN experimentation at arbitrarily large scales. GIST operates by dividing a global GCN model into several, disjoint subnetworks, training them independently, then combining their individual updates back into the global model.*

Background: What is a GCN?

- Deep learning works well for Euclidean data but not all data fits this representation
    - Many applications to rely on graph-structured data (e.g., social networks, chemical molecules, etc.)
- Several deep learning techniques have been extended into the graph domain, the most popular of which is the graph convolutional network (GCN)
    - GCN is a multi-layer architecture that implements a generalization of the convolution operation on graphs
    - GCN works very well for node/graph-level classification tasks
- **The GCN forward pass:**

Takeaway: GCNs are just MLPs with intermediate "aggregation" steps between neighboring nodes

Aggregate neighboring node representations

Input features: each row is a node feature vector

$$H_{\ell+1} = \sigma\left(\bar{A} H_\ell \Theta_\ell\right); \quad \text{where} \quad H_0 = X \quad \text{and} \quad \bar{A} = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$$

Non-linearity (e.g., ReLU)

Resembles and MLP by "mixing" node representations at each layer

Normalized adjacency matrix with added self-loops

Background: General

- GCNs are the most popular algorithm for deep learning in the graph domain, but their training is notoriously inefficient and scales poorly with graph/model size
  - The “receptive field” of a GCN expands exponentially with the depth (most GCNs have few layers for this reason and because of “over smoothing”)
  - Mini-batches (i.e., subsets of nodes in a graph) are not independent
- *How do we currently tackle this inefficiency?* Partition the graph into multiple sub-graphs that can be used as “mini-batches” during training
  - **Neighborhood sampling:** sample a subset of neighboring nodes at each GCN layer
    - E.g., FastGCN, LADIES, GraphSAGE
  - **Graph partitioning:** divide the graph into smaller sub-graphs for use during training
    - E.g., ClusterGCN, GraphSAINT
- The scale of GCN experimentation still lags behind that of deep learning!
  - Models are small (need more width!) and graph size is limited -- <u>we want to fix this</u>

Our proposal

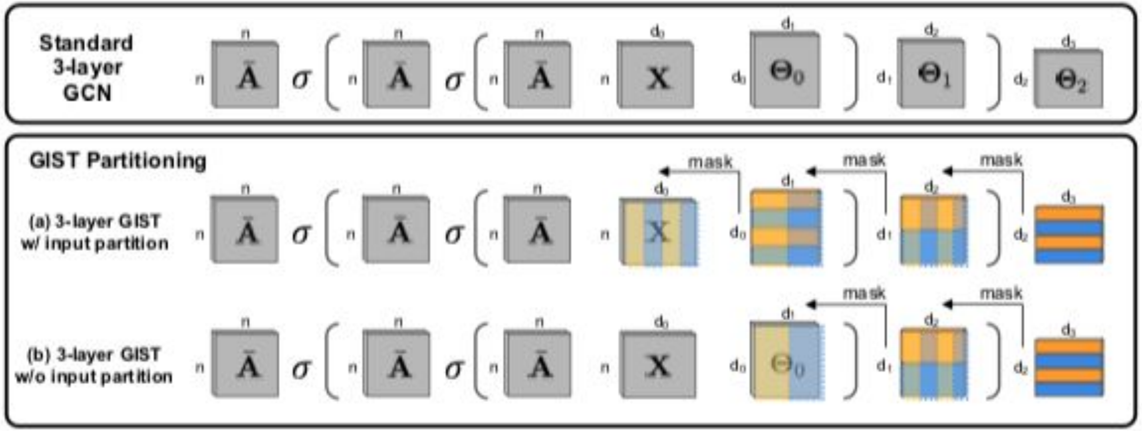
- GIST is a novel distributed training methodology that can be used for any GCN architecture and is compatible with neighborhood/graph sampling approaches
- **Methodology:**
  - Randomly decompose global GCN into disjoint, narrow sub-GCNs of equal depth
    - Randomly sample the feature space, each sub-GCNs has the same width
  - Train sub-GCNs independently and in parallel (on separate GPUs)
  - Copy sub-GCN parameters into the global model (disjoint partition prevents collisions)
  - Repeat until convergence
- GIST can be (optionally) combined with node partitioning to handle arbitrarily-large graphs
- **Benefits:** works for any GCN model, improved communication efficiency, improved training efficiency, lower wall-clock time from parallel training, ability to train “ultra-wide” models

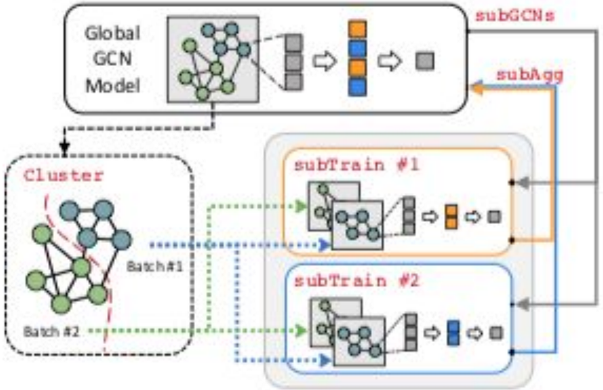
Our proposal

Algorithm 1 GIST Algorithm

Parameters: T synchronization iterations, m sub-GCNs, ζ local iterations, c clusters, $\mathcal{G}$ training graph.

$\Psi_{\mathcal{G}}(\cdot;\boldsymbol{\Theta}) \leftarrow$ randomly initialize GCN
$\{\mathcal{G}_{(j)}\}_{j=1}^{c} \leftarrow \texttt{Cluster}(\mathcal{G}, c)$
for $t = 0, \dots, T-1$ **do**
  $\{\Psi_{\mathcal{G}}(\cdot;\boldsymbol{\Theta}^{(i)})\}_{i=1}^{m} \leftarrow \texttt{subGCNs}(\Psi_{\mathcal{G}}(\cdot;\boldsymbol{\Theta}), m)$
  Distribute each $\Psi_{\mathcal{G}}(\cdot;\boldsymbol{\Theta}^{(i)})$ to a different worker
  for $i = 1, \dots, m$ **do**
    for $z = 1, \dots, \zeta$ **do**
      $\Psi_{\mathcal{G}}(\cdot;\boldsymbol{\Theta}^{(i)}) \leftarrow \texttt{subTrain}(\boldsymbol{\Theta}^{(i)}, \{\mathcal{G}_{(j)}\}_{j=1}^{c})$
    end for
  end for
  $\Psi_{\mathcal{G}}(\cdot;\boldsymbol{\Theta}) \leftarrow \texttt{subAgg}(\{\boldsymbol{\Theta}^{(i)}\}_{i=1}^{m})$
end for

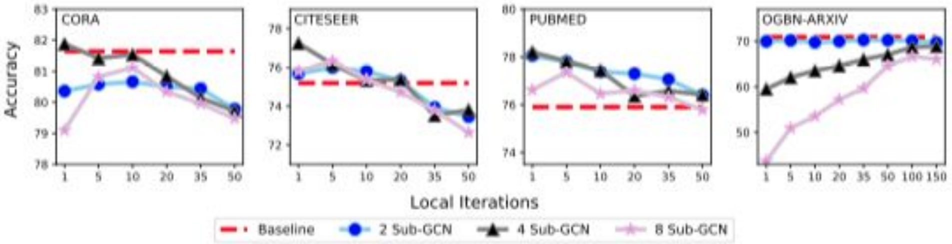




Small-Scale Experiments

- **Datasets:** Cora, Pubmed, Citeseer, OGBN-Arxiv
- **Experimental Settings:** 3-layer GCN, width 256, 400 epochs, Adam optimizer
- **Baseline:** identical model trained with standard, single-GPU methodology
- **Findings:** all features except the input can be split, performance is relatively robust to larger numbers of local iterations, performance is relatively robust as the number of sub-GCNs is increased, models trained with GIST often outperform single-GPU baseline models

# Sub-GCNs	d_0	d_1	d_2	Cora	Citeseer	Pubmed	OGBN-Arxiv
Baseline				81.52 ± 0.005	75.02 ± 0.018	75.90 ± 0.003	70.85 ± 0.089
2	✓	✓	✓	80.00 ± 0.010	$\mathbf{75.95} \pm 0.007$	76.68 ± 0.011	65.65 ± 0.700
	✓	✓		78.30 ± 0.011	69.34 ± 0.018	75.78 ± 0.015	65.33 ± 0.347
		✓	✓	$\mathbf{80.82} \pm 0.010$	75.82 ± 0.008	$\mathbf{78.02} \pm 0.007$	$\mathbf{70.10} \pm 0.224$
4	✓	✓	✓	76.78 ± 0.017	70.66 ± 0.011	65.67 ± 0.044	54.21 ± 1.360
	✓	✓		66.56 ± 0.061	68.38 ± 0.018	68.44 ± 0.014	52.64 ± 1.988
		✓	✓	$\mathbf{81.18} \pm 0.007$	$\mathbf{76.21} \pm 0.017$	$\mathbf{76.99} \pm 0.006$	$\mathbf{68.69} \pm 0.579$
8	✓	✓	✓	48.32 ± 0.087	45.42 ± 0.092	54.29 ± 0.029	40.26 ± 1.960
	✓	✓		53.60 ± 0.020	54.68 ± 0.030	51.44 ± 0.002	26.84 ± 7.226
		✓	✓	$\mathbf{79.58} \pm 0.006$	$\mathbf{75.39} \pm 0.016$	$\mathbf{76.99} \pm 0.006$	$\mathbf{65.81} \pm 0.378$



Large-Scale Experiments

- **Datasets:** Reddit, Amazon2M
- **Experimental Settings:**
  - *Reddit:* GraphSage and GAT architectures, width 256, 2-4 layers, 500 local iterations, partition graph into 15,000 sub-graphs (METIS), batch size 10, Adam optimizer, 80 epochs, 2-8 sub-GCNs
  - *Amazon2M:* GraphSAGE architecture, width 400 or 4096, 5K local iterations, partition graph into 15,000 sub-graphs (METIS), batch size of 10, Adam optimizer, 400 epochs, 2-8 sub-GCNs
- **Baseline:** identical model trained with standard, single-GPU methodology
- **Metrics:** performance is measured as (i) wall-clock time and (ii) F1 classification score

Large-Scale Experiments

- **Findings on Reddit:** GIST significantly accelerates GCN training and sometimes improves accuracy, more sub-GCNs slightly degrades accuracy but improves wall-clock time, speedup from GIST is more significant for models/datasets with larger compute requirements (e.g., GAT vs. GraphSAGE, Amazon2M vs. Reddit).

L	# Sub-GCNs	GraphSAGE			GAT		
		F1 Score	Time	Speedup	F1 Score	Time	Speedup
2	Baseline	96.09	105.78s	1.00×	89.57	4289.80s	1.00×
	2	96.40	70.29s	1.50×	90.28	2097.16s	2.05×
	4	96.16	68.88s	1.54×	90.02	1112.33s	3.86×
	8	95.46	76.68s	1.38×	89.01	640.44s	6.70×
3	Baseline	96.32	118.37s	1.00×	89.25	7241.23s	1.00×
	2	96.36	80.46s	1.47×	89.63	3432.80s	2.11×
	4	95.76	78.74s	1.50×	88.82	1727.73s	4.19×
	8	94.39	88.54s	1.34×	70.38	944.21s	7.67×
4	Baseline	96.32	120.74s	1.00×	88.36	9969.13s	1.00×
	2	96.01	91.75s	1.32×	87.97	4723.55s	2.11×
	4	95.21	78.74s	1.53×	78.42	2376.21s	4.21×
	8	92.75	88.71s	1.36×	66.30	1262.28s	7.90×

Large-Scale Experiments

- **Findings on Amazon2M:** narrow models perform worse than the baseline but train much faster, a more significant speedup is observed for wide models (up to >9X), GIST performs better (relative to baseline) with wider models, baseline models take longer to achieve comparable F1 score in comparison to models trained with GIST (e.g., for L=2 and m=8, standard training takes 2.5X longer to achieve F1 score of 88.86)

L	# Sub-GCNs	$d_i = 400$			$d_i = 4096$		
		F1 Score	Time	Speedup	F1 Score	Time	Speedup
2	Baseline	89.90	1.81hr	1.00×	91.25	5.17hr	1.00×
	2	88.36	1.25hr	1.45×	90.70	1.70hr	3.05×
	4	86.33	1.11hr	1.63×	89.49	1.13hr	4.57×
	8	84.73	1.13hr	1.61×	88.86	1.11hr	4.65×
3	Baseline	90.36	2.32hr	1.00×	91.51	9.52hr	1.00×
	2	88.59	1.56hr	1.49×	91.12	2.12hr	4.49×
	4	86.46	1.37hr	1.70×	89.21	1.42hr	6.72×
	8	84.76	1.37hr	1.69×	86.97	1.34hr	7.12×
4	Baseline	90.40	3.00hr	1.00×	91.61	14.20hr	1.00×
	2	88.56	1.79hr	1.68×	91.02	2.77hr	5.13×
	4	87.53	1.58hr	1.90×	89.07	1.65hr	8.58×
	8	85.32	1.56hr	1.93×	87.53	1.55hr	9.13×

Ultra-Wide GCN Models on Amazon2M

- **Ultra-Wide GCN findings:** GIST can train models to SOTA performance with width up to 32K (exceeds capacity of single GPU by ~8X), best performance (91.09 F1 score) is achieved by model with width 16K, standard training methodologies are very limited in model width, more sub-GCNs provide massive speedups as model width increases (e.g., 18X speedup with 2 vs. 8 sub-GCNs with 32K width when L=2), single-GPU models take longer to train cannot match F1 score of ultra-wide models trained with GIST .

L	# Sub-GCNs	F1 Score (Time)				
		$d_i = 400$	$d_i = 4096$	$d_i = 8192$	$d_i = 16384$	$d_i = 32768$
2	Baseline	89.38 (1.81hr)	90.58 (5.17hr)	OOM	OOM	OOM
	2	87.48 (1.25hr)	90.09 (1.70hr)	90.87 (2.76hr)	90.94 (9.31hr)	90.91 (32.31hr)
	4	84.82 (1.11hr)	88.79 (1.13hr)	89.76 (1.49hr)	90.10 (2.24hr)	90.17 (5.16hr)
	8	82.56 (1.13hr)	87.16 (1.11hr)	88.31 (1.20hr)	88.89 (1.39hr)	89.46 (1.76hr)
3	Baseline	89.73 (2.32hr)	90.99 (9.52hr)	OOM	OOM	OOM
	2	87.79 (1.56hr)	90.40 (2.12hr)	90.91 (4.87hr)	91.05 (17.7hr)	OOM
	4	85.30 (1.37hr)	88.51 (1.42hr)	89.75 (2.07hr)	90.15 (3.44hr)	OOM
	8	82.84 (1.37hr)	86.12 (1.34hr)	88.38 (1.37hr)	88.67 (1.88hr)	88.66 (2.56hr)
4	Baseline	89.77 (3.00hr)	91.02 (14.20hr)	OOM	OOM	OOM
	2	87.75 (1.79hr)	90.36 (2.77hr)	91.08 (6.92hr)	91.09 (26.44hr)	OOM
	4	85.32 (1.58hr)	88.50 (1.65hr)	89.76 (2.36hr)	90.05 (4.93hr)	OOM
	8	83.45 (1.56hr)	86.60 (1.55hr)	88.13 (1.61hr)	88.44 (2.30hr)	OOM

Conclusions

- GIST is a distributed training approach for GCNs that enables exploration of larger models and datasets while minimizing the wall-clock time of experiments
    - Compatible with any GCN architecture
    - Can be easily combined with existing sampling approaches for efficient GCN training
- GIST uses a feature-wise partition of a global GCN into several sub-GCNs, which are trained independently and in parallel before having their updates aggregated
- GIST achieves remarkable speed-ups on large-scale GCN experiments and enables the training of ultra-wide GCNs that cannot be handled with standard training methodology
- **GIST is a step towards exploring experimental scales comparable to deep learning within the graph ML community**