Decoupled Greedy Learning of Graph Neural Networks

Yewen Wang*, Jian Tang+, Yizhou Sun*, Wolf Guy+
*: Department of Computer Science, University of California, Los Angeles, +: MILA

Motivation

Improve the efficiency of GNN training:

- Problem 1: Recursive computation
- Problem 2: Update-locking
  - each layer heavily relies on upper layers' feedback to up-date itself
  - it must wait for the information to propagate through the whole network before updating

Main Contributions

- Introduce a decoupled greedy learning algorithm for GNNs
  - achieves update-unlocking
  - enables GNN layers to be trained in parallel
    - Less time, less per-GPU memory, good for time/hardware-limited applications
- Leverage a lazy-update scheme
  - Further improves efficiency
- Our method can be used in more general cases:
  - not limited to the deep GCN model
  - not limited to node classification task
  - can be combined with other scalability-enhancing GNNs and can be applied to other graph-related tasks

Conventional GNN and Layer-wise GNN

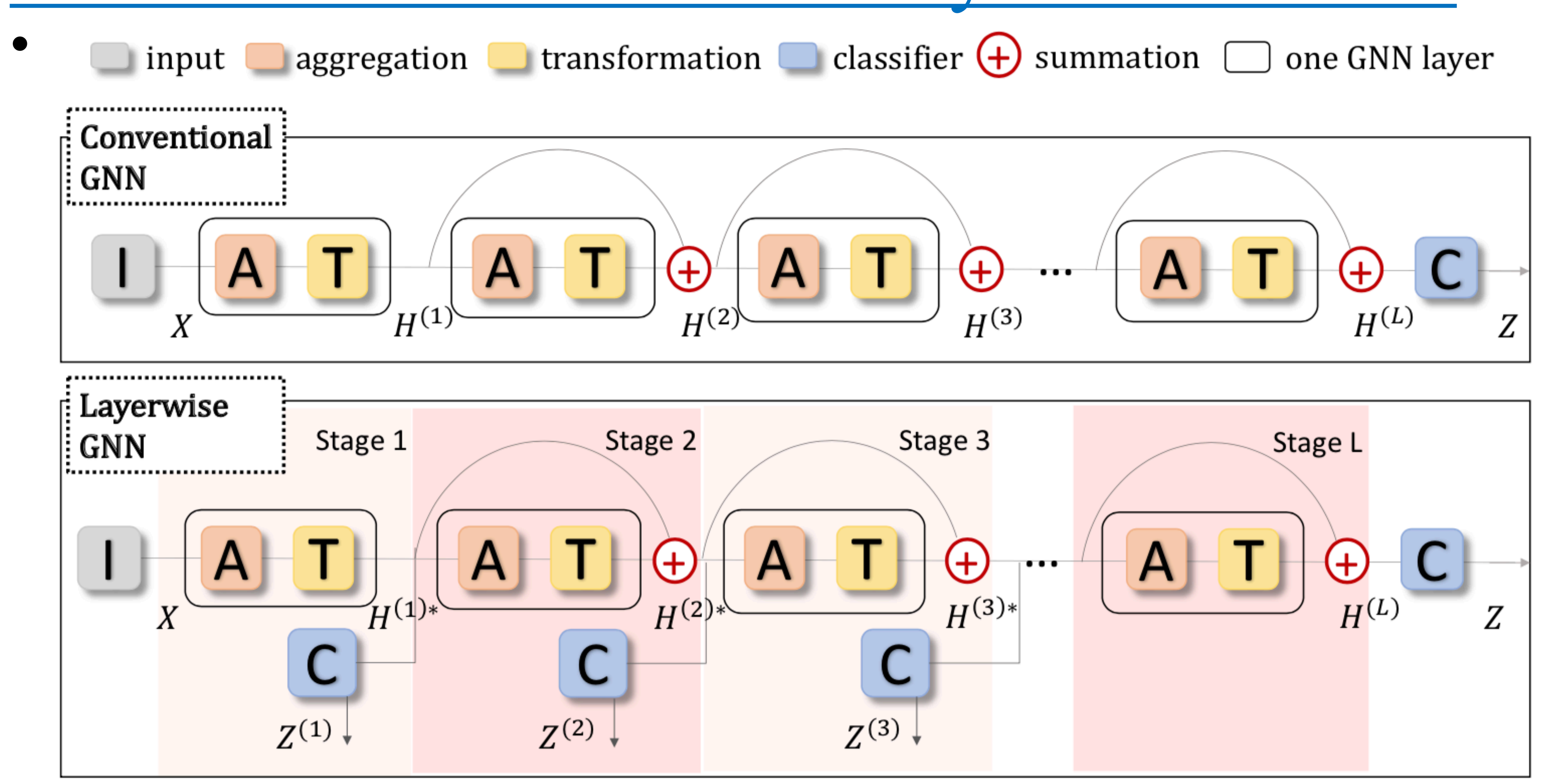

Fig1. High level framework of conventional GNN (upper) and layer-wise GNN. The aggregation step (A) corresponds to $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l-1)}$operation and the transformation step corresponds to $\sigma(\cdot\, W^{(l)})$operation

Decoupled Greedy Learning GNN

- Method:
  - Decouple the GNN into different layers, append one auxiliary greedy objective (node classification) after each layer, and enable parallelization.
  - Leverage the Lazy Update scheme to improve efficiency.
  - Analogy: block coordinate descent method.
- Main Algorithms

Algorithm 1 Decoupled Greedy Learning (DGL) of GNNs

Require: Normalized Adjacency Matrix $\boldsymbol{F}$; Feature Matrix $\boldsymbol{X}$; Labels $\boldsymbol{Y}$; Total Number of Iterations T; Total Number of Layers L.

Initialize: $\boldsymbol{H}^{(0)} = \boldsymbol{X}$;
for $t = 1$ to T **do**
  for $l = 1$ to L **do**
    $\boldsymbol{H}^{(l)} = \sigma(\boldsymbol{F}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})$ *// Get node embeddings and store them as* $\boldsymbol{H}^{(l)}$.
    $(\boldsymbol{W}^{(l)}, \Theta^{(l)}) \leftarrow$ Update with $\nabla loss_{(\boldsymbol{W}^{(l)},\Theta^{(l)})}(\boldsymbol{Y}, \boldsymbol{H}^{(l-1)}, \boldsymbol{F}; \boldsymbol{W}^{(l)}, \Theta^{(l)})$ *// Update parameters.*
  end for
end for

Algorithm 2 Decoupled Greedy Learning (DGL) of GNNs with Lazy Update Scheme

Require: Normalized Adjacency Matrix $\boldsymbol{F}$; Feature Matrix $\boldsymbol{X}$; Labels $\boldsymbol{Y}$; Total Number of Iterations T; Total Number of Layers L; Waiting time T_{lazy}.

Initialize: $\hat{\boldsymbol{H}}^{(0)} = \boldsymbol{F}\boldsymbol{X}$;
for $t = 1$ to T **do**
  for $l = 1$ to L **do**
    $\boldsymbol{H}^{(l)} = \sigma(\hat{\boldsymbol{H}}^{(l-1)}\boldsymbol{W}^{(l)})$ *// Get node embeddings.*
    $(\boldsymbol{W}^{(l)}, \Theta^{(l)}) \leftarrow$ Update with $\nabla loss_{(\boldsymbol{W}^{(l)},\Theta^{(l)})}(\boldsymbol{Y}, \hat{\boldsymbol{H}}^{(l-1)}; \boldsymbol{W}^{(l)}, \Theta^{(l)})$ *// Update parameters.*
    if $(t \bmod T_{lazy} == 0)$ **then**
      $\hat{\boldsymbol{H}}^{(l)} = \boldsymbol{F}\boldsymbol{H}^{(l)}$ *// Get propagated node embeddings and store them as* $\hat{\boldsymbol{H}}^{(l)}$.
    end if
  end for
end for

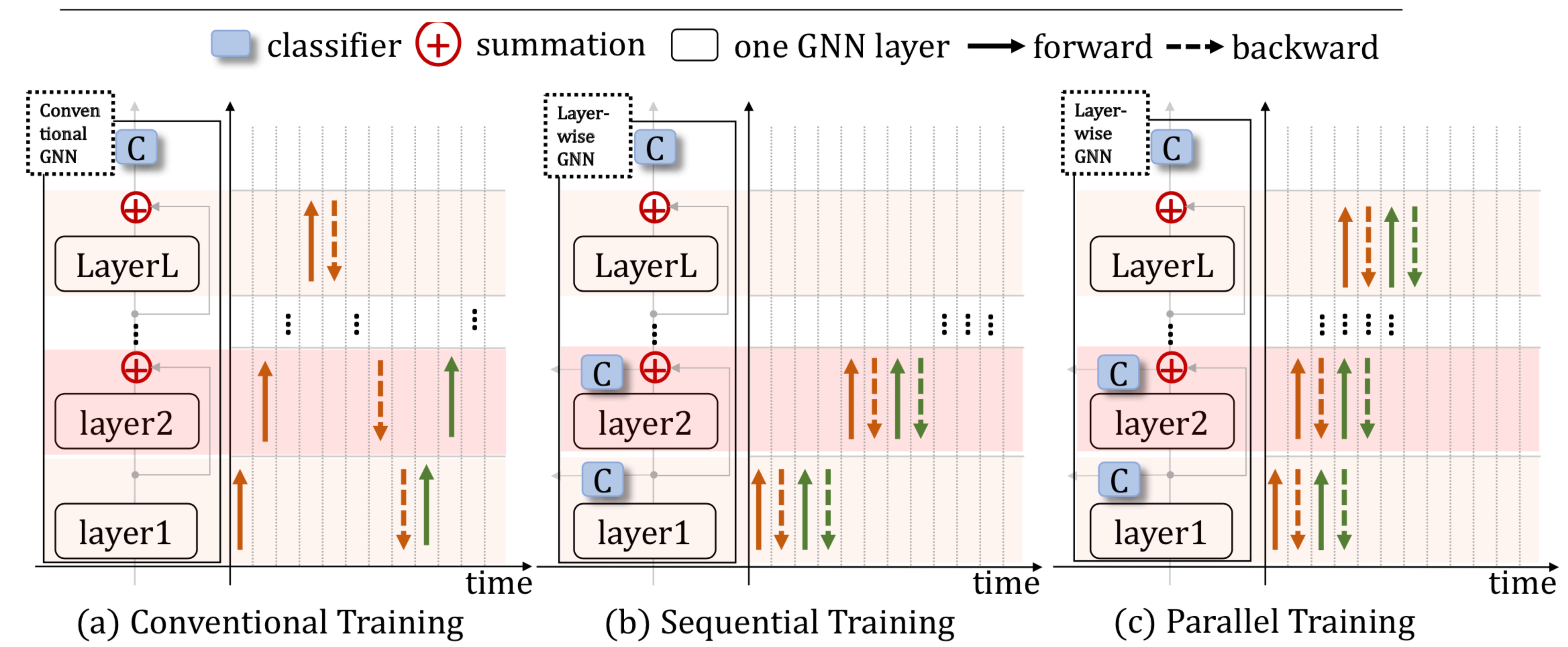

Fig2. Signal propagation process for 3 GNN training methods: Conventional joint training, Sequential layer-wise training, Parallel layer-wise training. Arrows of different colors represents different batches of data.

Complexity Comparison

Methods	Memory (per GPU)	Time
Full-Batch GCN	$\mathcal{O}(LNK + LK^2)$	$\mathcal{O}(TL\Vert\boldsymbol{A}\Vert_0 K + TLNK^2)$
GraphSage	$\mathcal{O}(bKs_{node}^{L-1} + LK^2)$	$\mathcal{O}(bTKs_{node}^{L} + bTK^2 s_{node}^{L-1})$
VR-GCN	$\mathcal{O}(LNK + LK^2)$	$\mathcal{O}(b\bar{D}TKs_{node}^{L-1} + bTK^2 s_{node}^{L-1})$
FastGCN	$\mathcal{O}(LKs_{layer} + LK^2)$	$\mathcal{O}(TLKs_{layer}^2 + TLK^2 s_{layer})$
LADIES	$\mathcal{O}(LKs_{layer} + LK^2)$	$\mathcal{O}(TLKs_{layer}^2 + TLK^2 s_{layer})$
ClusterGCN	$\mathcal{O}(bLK + LK^2)$	$\mathcal{O}(TL\Vert\boldsymbol{A}\Vert_0 K + TLNK^2)$
L2GCN	$\mathcal{O}(NK + 2K^2)$	$\mathcal{O}(L\Vert\boldsymbol{A}\Vert_0 K + 2TLNK^2)$
LU-DGL-GCN (ours)	$\mathcal{O}(NK + 2K^2)$	$\mathcal{O}(T\Vert\boldsymbol{A}\Vert_0 K/T_{wait} + 2TNK^2)$

Tab.1 Summary of Complexity. $\overline{D}$ is the avg degree, b is the batch size, s_{node} and s_{layer} are the num of sampled neighbors the sampling-based baselines, K is the dim of embedding vectors, L is the num of layers, N is the num of nodes in the graph, A is the adj matrix, T is the num of iterations, T_{wait} is the waiting time for LU-DGL-GCN.

Results

	cora			citeseer			pubmed		
	acc	mem	time	acc	mem	time	acc	mem	time
GCN	77.8 ± 1.3	31.7	42.2 ± 1.0	65.5 ± 2.4	67.9	33.1 ± 1.2	74.8 ± 2.6	137.9	46.9 ± 2.0
GIN	77.0 ± 1.1	31.7	37.7 ± 1.2	65.8 ± 1.2	67.9	37.1 ± 1.6	75.4 ± 2.3	137.9	46.3 ± 2.0
LADIES(64)	78.8 ± 0.8	3.1	31.5 ± 0.8	66.6 ± 1.2	5.9	32.5 ± 1.3	77.9 ± 2.4	1.9	33.9 ± 1.5
LADIES(512)	79.8 ± 1.5	7.4	32.3 ± 0.8	66.8 ± 3.6	13.9	35.9 ± 1.2	78.3 ± 0.9	4.4	38.3 ± 1.6
FastGCN(64)	55.3 ± 4.8	3.1	36.8 ± 2.1	35.9 ± 1.0	5.9	34.5 ± 1.7	41.2 ± 0.5	1.9	34.6 ± 1.4
FASTGCN(512)	79.6 ± 1.4	7.4	37.1 ± 1.7	66.7 ± 1.4	13.9	35.8 ± 1.8	76.7 ± 1.2	4.5	37.4 ± 2.4
LGCN	80.4 ± 0.9	6.9	119.6 ± 11.5	67.1 ± 1.7	14.7	107.9 ± 5.8	76.2 ± 1.6	29.1	141.8 ± 12.8
LU-DGL-GCN(50)	78.0 ± 1.3	6.9	14.1 ± 0.4	64.8 ± 6.3	14.7	13.9 ± 0.2	76.9 ± 5.3	29.2	14.9 ± 1.0
LGIN	81.1 ± 1.3	6.9	80.9 ± 2.2	66.6 ± 1.1	14.7	84.7 ± 2.2	76.7 ± 1.4	--	97.3 ± 2.3
LU-DGL-GIN(1)	80.0 ± 0.6	6.9	14.0 ± 0.2	55.2 ± 3.9	14.7	14.0 ± 0.1	77.5 ± 0.3	--	15.9 ± 0.2
LLADIES(64)	80.4 ± 0.8	0.7	55.5 ± 0.9	66.5 ± 1.2	1.3	59.5 ± 1.4	78.4 ± 0.6	0.4	66.7 ± 0.7
LLADIES(512)	80.6 ± 1.1	1.6	101.5 ± 3.1	68.9 ± 1.0	2.8	96.6 ± 3.6	76.7 ± 0.7	0.9	103.1 ± 3.5
LU-DGL-LADIES(64,1)	77.4 ± 1.4	0.8	13.2 ± 0.2	50.0 ± 1.4	1.3	13.7 ± 0.5	76.8 ± 1.5	0.4	14.7 ± 0.3
LU-DGL-LADIES(512,1)	80.0 ± 0.6	1.6	13.7 ± 0.5	54.3 ± 4.2	2.9	13.8 ± 0.1	77.5 ± 1.2	1.0	14.8 ± 0.3

- *Compare GCN, LGCN, LU-DGL-GCN*: Our method is very efficient , it can save time and per-GPU memory without too much compromising on performance.
- *Compare GIN, LGIN, LU-DGL-GIN*: Our method is not limited to GCN but can be combined with other GNN models.
- *Compare LADIES, LLADIES, LU-DGL-LADIES*: The proposed method can be combined with other scalability-enhancing methods for GNNs.

Acknowledgement

This work was partially funded by IVADO Professor startup & operational funds, IVADO Fundamental Research Project grant PRF-2019-3583139727 [*G.W.*], NSF III-1705169, NSF CAREER Award 1741634, NSF #1937599, DARPA HR00112090027, Okawa Foundation Grant, and Amazon Research Award *[Y.S.]*.