Scaling GNNs: Efficient models, Sampling models

I. Makarov & V. Pozdnyakov & D. Kiselev

BigData Academy MADE from Mail.ru Group

Graph Neural Networks and Applications

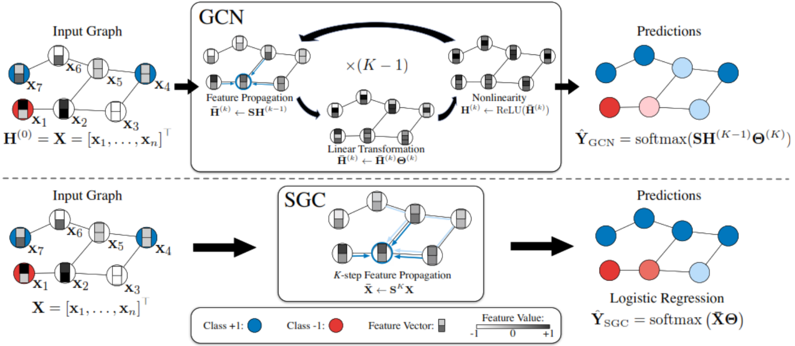
Topics

1. Efficient training
2. Finding proper architectures
3. Graph coarsening and large-scale training
4. Sampling GNN models

Efficient training

Simplifying Graph Convolutional Networks

1. How to solve oversmoothing problem with sparse labels?
2. Let's approximate K-layer GCN with A^K feature propagation



from Weinberger et al., 2019

Adaptive Label Smoothing To Regularize Large-Scale Graph Training

1. Adapt label smoothing via two-stage PageRank approximation
2. Consistent labels between prediction and ground truth, propagation and ground truth, and regularization on uniform distribution for propagation

$$\mathcal{L}^{LS}(\theta) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} H(\boldsymbol{y}_i^{LS}, \hat{\boldsymbol{y}}_i) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}(1-\alpha)H(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) + \alpha H(\mathbf{1}/C, \hat{\boldsymbol{y}}_i)$$

$$\boldsymbol{Y}^{(k+1)} = (1-\beta)\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{Y}^{(k)} + \beta\boldsymbol{Y}^{(0)} \qquad \boldsymbol{y}_i^{\text{soft}} = \text{Softmax}(\boldsymbol{W}\boldsymbol{y}_i^{(K)})$$

$$\boldsymbol{y}_i^{ALS} = (1-\alpha)\boldsymbol{y}_i + \alpha\boldsymbol{y}_i^{\text{soft}}$$

$$\begin{aligned}\mathcal{L}^{ALS}(\theta, \boldsymbol{W}) &= \tfrac{1}{|\mathcal{B}|}\textstyle\sum_{i\in\mathcal{B}} H(\boldsymbol{y}_i^{ALS}, \hat{\boldsymbol{y}}_i) + \gamma\text{KL}(\boldsymbol{y}_i^{\text{soft}}, \mathbf{1}/C) \\ &= \tfrac{1}{|\mathcal{B}|}\textstyle\sum_{i\in\mathcal{B}}(1-\alpha)H(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) + \alpha H(\boldsymbol{y}_i^{\text{soft}}, \hat{\boldsymbol{y}}_i) + \gamma\text{KL}(\boldsymbol{y}_i^{\text{soft}}, \mathbf{1}/C)\end{aligned}$$

from Hu et al., 2021

Adaptive Filters and Aggregator Fusion for Efficient Graph Convolutions

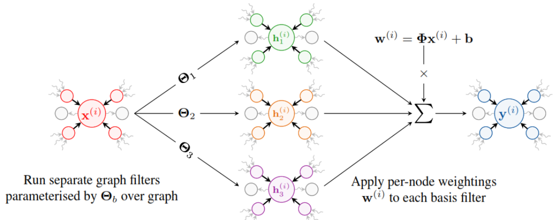
1. Different aggregation and fusion schema

Method	Propagation Rule	Memory	Notes
GCN (Kipf & Welling, 2017)	$\mathbf{y}^{(i)} = \mathbf{\Theta} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)\deg(j)}} \mathbf{x}^{(j)}$	$\mathcal{O}(\lvert V \rvert)$	Formally defined for undirected graphs with self-loops; motivated by graph signal processing.
GAT (Veličković et al., 2018)	$\mathbf{y}^{(i)} = \alpha_{i,i} \mathbf{\Theta} \mathbf{x}^{(i)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \mathbf{\Theta} \mathbf{x}^{(j)}$	$\mathcal{O}(\lvert E \rvert)$	Attention coefficients calculated using dot-product attention: $\alpha_{i,j} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}^{(i)} \parallel \mathbf{\Theta}\mathbf{x}^{(j)}]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}^{(i)} \parallel \mathbf{\Theta}\mathbf{x}^{(k)}]\right)\right)}$. Common to define multiple attention heads and concatenate.
GIN (Xu et al., 2019)	$\mathbf{y}^{(i)} = f_{\mathbf{\Theta}}[(1+\epsilon)\mathbf{x}^{(i)} + \sum_{j \in \mathcal{N}(i)} \mathbf{x}^{(j)}]$	$\mathcal{O}(\lvert V \rvert)$	f is a learnable function, typically parameterized as an MLP or linear layer; ϵ may be fixed or learned.
MPNN (Gilmer et al., 2017)	$\mathbf{y}^{(i)} = U(\mathbf{x}^{(i)}, \bigoplus_{j \in \mathcal{N}(i)} M(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{e}_{ij}))$	$\mathcal{O}(\lvert E \rvert)$	U, M typically defined as linear layers acting on concatenated features; $\bigoplus$ may be any valid aggregator, typically sum or max.
PNA (Corso et al., 2020)	$\mathbf{y}^{(i)} = U(\mathbf{x}^{(i)}, \bigoplus_{j \in \mathcal{N}(i)} M(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{e}_{ij}))$	$\mathcal{O}(\lvert E \rvert)$	Similar to MPNN, but with $\bigoplus$ defined to use 4 aggregators (mean, standard deviation, max, and min) scaled by 3 different functions of node degree, resulting in 12 different aggregations.

from Lane et al., 2021

Adaptive Filters and Aggregator Fusion for Efficient Graph Convolutions

1. General model for GCN, GAT and GraphSAGE/GIN



$$\mathbf{y}^{(i)} = \sum_{b=1}^{B} w_b^{(i)} \sum_{j \in \mathcal{N}(i)} \alpha(i,j)\mathbf{\Theta}_b \mathbf{x}^{(j)} \qquad \mathbf{y}^{(i)} = \mathop{\Big\Vert}_{h=1}^{H} \sum_{b=1}^{B} w_{h,b}^{(i)} \sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{\deg(i)\deg(j)}} \mathbf{\Theta}_b \mathbf{x}^{(j)}$$

$$\mathbf{y}^{(i)} = \mathop{\Big\Vert}_{h=1}^{H} \sum_{\oplus \in \mathcal{A}} \sum_{b=1}^{B} w_{h,\oplus,b}^{(i)} \bigoplus_{j \in \mathcal{N}(i)\cup\{i\}} \mathbf{\Theta}_b \mathbf{x}^{(j)}$$

EGC-S

$$\mathbf{y}^{(i)} = \mathop{\Big\Vert}_{h=1}^{H} \left(\sum_{b=1}^{B} w_{h,b}^{(i)} \mathbf{\Theta}_b\right) \left(\sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{\mathbf{x}^{(j)}}{\sqrt{\deg(i)\deg(j)}}\right)$$

$$= \mathop{\Big\Vert}_{h=1}^{H} \underbrace{\mathbf{\Theta}_h^{(i)}}_{\text{Varying per Node}} \underbrace{\left(\sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{\mathbf{x}^{(j)}}{\sqrt{\deg(i)\deg(j)}}\right)}_{\text{Computable via SpMM}}$$

GAT

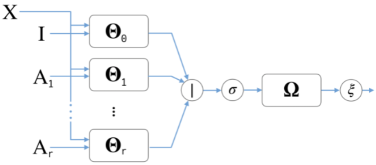
$$\mathbf{y}^{(i)} = \mathop{\Big\Vert}_{h=1}^{H} \alpha_{h,i,i} \mathbf{\Theta}\mathbf{x}^{(i)} + \sum_{j \in \mathcal{N}(i)} \alpha_{h,i,j} \mathbf{\Theta}\mathbf{x}^{(j)}$$

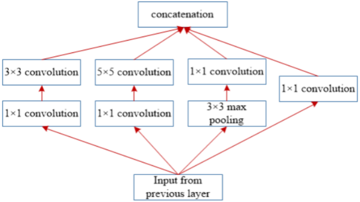
$$= \mathop{\Big\Vert}_{h=1}^{H} \underbrace{\mathbf{\Theta}}_{\text{Shared Weights}} \underbrace{\left(\sum_{j \in \mathcal{N}(i)\cup\{i\}} \alpha_{h,i,j} \mathbf{x}^{(j)}\right)}_{\text{Message Materialization}}$$

from Lane et al., 2021

SIGN: Scalable Inception Graph Neural Networks

1. General simplified GCN and feature propagation under unified framework similar to Inception block in CNNs





$$\mathbf{Y} = \xi(\tilde{\mathbf{A}}^L \mathbf{X} \mathbf{\Theta}^{(1)} \cdots \mathbf{\Theta}^{(L)}) = \xi(\tilde{\mathbf{A}}^L \mathbf{X} \mathbf{\Theta})$$

from Monti et al., 2020

Training Graph Neural Networks with 1000 Layers

1. Use Residual blocks with post-normalization to remove vanishing/exploding gradient problem

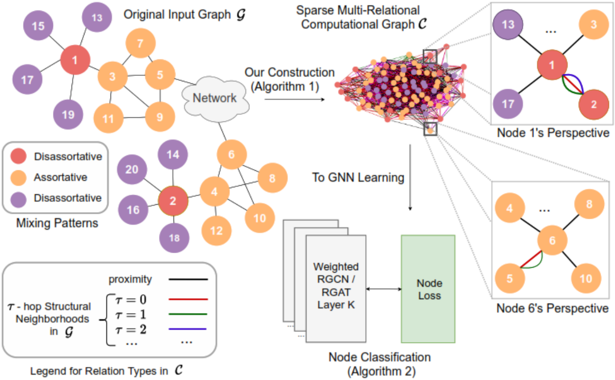
$$\langle X_1, X_2, ..., X_C \rangle$$
$$X_0' = \sum_{i=2}^{C} X_i$$
$$X_i' = f_{w_i}(X_{i-1}', A, U) + X_i,\ i \in \{1, \cdots, C\}$$
$$\widehat{X}_i = \text{Dropout}(\text{ReLU}(\text{Norm}(X_{i-1}')))$$
$$\widetilde{X}_i = \text{GraphConv}(\widehat{X}_i, A, U).$$

$$Z' = \text{GraphConv}(Z_{\text{in}}, A, U)$$
$$Z^{''} = \text{Norm}(Z' + X)$$
$$Z^{'''} = \text{GraphConv}(\text{Dropout}(\text{ReLU}(Z^{''})), A, U)$$
$$Z_{\text{o}} = \text{Norm}(\text{ReLU}(Z^{'''} + Z')),$$

from Koltun et al., 2021

Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns

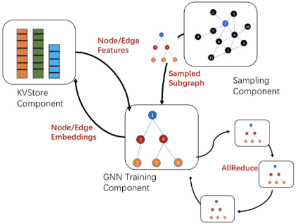
1. Model uses both proximity and structural information to build a computation graph on which a GNN is run

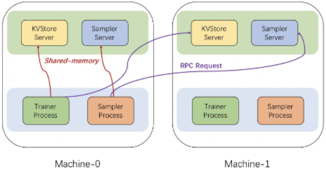


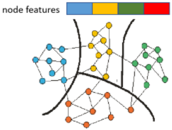
from Ma et al., 2021

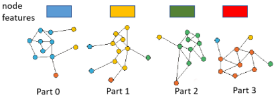
DistDGL: Distributed Graph Neural Network Training for Billion-Scale Graphs

1. Distributed DGL (or e.g., DistGNN), smart partitions with HALO vertices







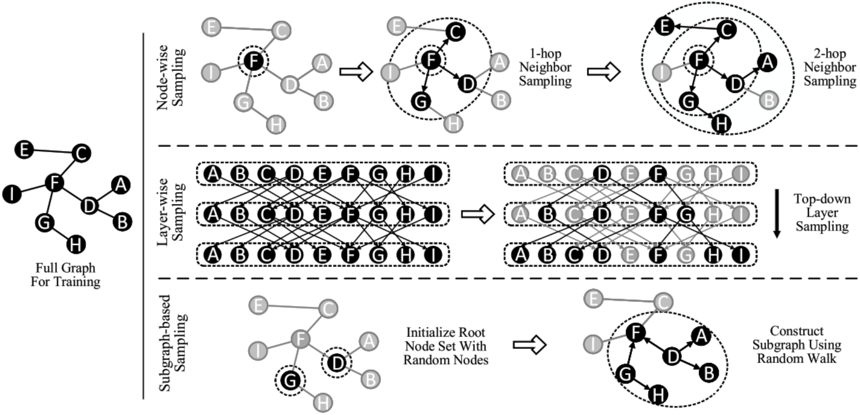


from Karypis et al., 2020

Graph Sampling

Training GNN with Sampling

1. Sample neighborhood
2. Aggregate message
3. Combine with self representation



from Fan et al., 2021

Training GNN with Sampling

1. Various sampling strategies

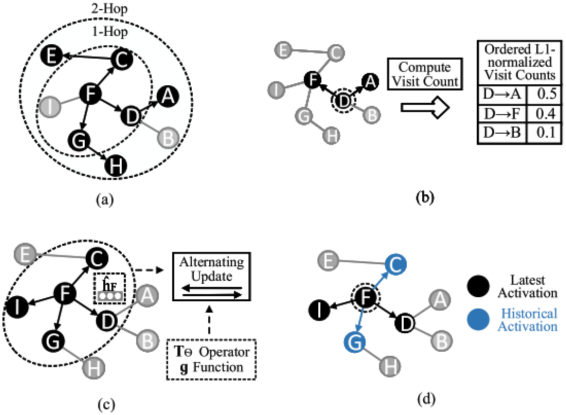
TABLE I: Categories and correlative works of the sampling methods

Categories	Works
Node-wise sampling method	GraphSAGE [35], PinSage [44], SSE [47], VR-GCN [48]
Layer-wise sampling method	FastGCN [49], AS-GCN [50], LADIES [51]
Subgraph-based sampling method	Cluster-GCN [52], GraphSAINT [53], RWT [54], Parallelized Graph Sampling [55]
Heterogeneous sampling method	Time-related sampling [56], HetGNN [57], HGSampling [58], Text Graph Sampling [59]

from Fan et al., 2021

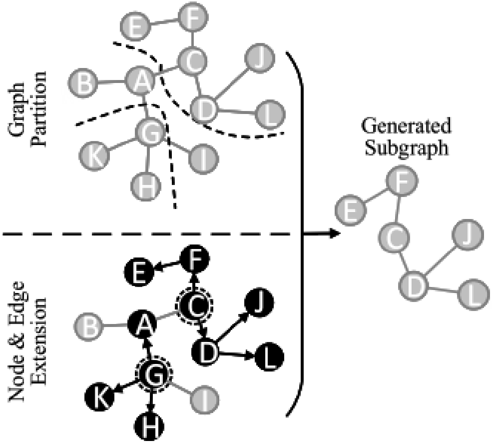
Training GNN with Sampling

1. (a) 2-hop neighbor sampling in GraphSAGE.
2. (b) Importance-based neighbor sampling in PinSage.
3. (c) Alternating sampling for embedding and update in SSE.
4. (d) Neighbor sampling leveraging historical activation in VR-GCN.



Training GNN with Sampling

1. Subgraph-based sampling



from Fan et al., 2021

Training GNN with Sampling

1. Heterogeneous sampling

TABLE VIII: Summary of the comparisons between heterogeneous sampling methods

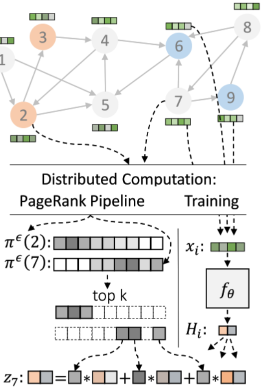
Method	**Sampling Target**	**Sampling Condition**	**Extra Trick**
Time-related Sampling [56]	Comments (Edges)	Sample the Closest Comments in terms of Publish Time	Use of Padded Placeholders
HetGNN [57]	Nodes in (random) All Types	Sample the Most Frequently Visited Nodes in RWR(v)	Random Walk with Restart Grouping by Types
HGSampling [58]	Nodes in (ordered) All Types	Sampling Probability Various in Different Node Types	Budget Stores Nodes by Types
Text Graph Sampling [59]	Document Nodes and Word Nodes	Sample the Most Intimate Nodes by Types	Strategy to Calculate the Intimacy Matrix

from Fan et al., 2021

Sampling Models

Scaling Graph Neural Networks with Approximate PageRank

1. Approximate PageRank
2. Aggregate only from top-k PageRank (similar to PinSAGE)



Algorithm 1 Approximate personalized PageRank (G, α, t, ϵ) [5]

Inputs: Graph G, teleport prob. α, target node t, max. residual ϵ

Initialize the (sparse) estimate-vector $\boldsymbol{\pi}^{(\epsilon)} = \mathbf{0}$ and the (sparse) residual-vector $\boldsymbol{r} = \alpha \cdot \boldsymbol{e}_t$ (i.e. $\boldsymbol{e}_t = 1, \boldsymbol{e}_v = 0, v \neq t$)

while $\exists v\, s.t.\, \boldsymbol{r}_v > \alpha \cdot \epsilon \cdot \boldsymbol{d}_v$ **do** # $\boldsymbol{d}_v$ is the out-degree

  $\boldsymbol{\pi}_v^{(\epsilon)}$ += $\boldsymbol{r}_v$

  $\boldsymbol{r}_v = 0$

  $m = (1 - \alpha) \cdot \boldsymbol{r}_v / \boldsymbol{d}_v$

  for $u \in N_G^{\text{out}}(v)$ **do** # v's outgoing neighbors

    $\boldsymbol{r}_u$ += m

  end for

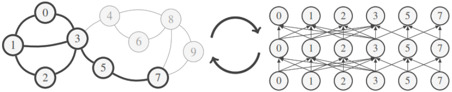
end while

return $\boldsymbol{\pi}^{(\epsilon)}$

from Günnemann et al., 2020

GraphSAINT

1. Efficient sampling via edge covering and node-edge estimation to co-occur in sample
2. Unbiased estimator for feature propagation
3. Ability to implement residual/attention architectures



$\mathcal{G}_s = \texttt{SAMPLE}(\mathcal{G})$ Full GCN on $\mathcal{G}_s$

Algorithm 1 `GraphSAINT` training algorithm

Input: Training graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$; Labels $\overline{\boldsymbol{Y}}$; Sampler `SAMPLE`;
Output: GCN model with trained weights

Pre-processing: Setup `SAMPLE` parameters; Compute normalization coefficients α, λ.
for each minibatch **do**
  $\mathcal{G}_s(\mathcal{V}_s, \mathcal{E}_s) \leftarrow$ Sampled sub-graph of $\mathcal{G}$ according to `SAMPLE`
  GCN construction on $\mathcal{G}_s$.
  $\{\boldsymbol{y}_v \mid v \in \mathcal{V}_s\} \leftarrow$ Forward propagation of $\{\boldsymbol{x}_v \mid v \in \mathcal{V}_s\}$, normalized by α
  Backward propagation from λ-normalized loss $L(\boldsymbol{y}_v, \overline{\boldsymbol{y}}_v)$. Update weights.
end for

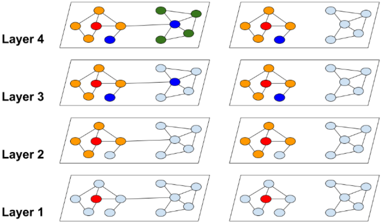
$$\zeta_v^{(\ell+1)} = \sum_{u \in \mathcal{V}} \frac{\widetilde{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} \left(\boldsymbol{W}^{(\ell)}\right)^{\mathsf{T}} \boldsymbol{x}_u^{(\ell)} \mathbb{1}_{u|v} = \sum_{u \in \mathcal{V}} \frac{\widetilde{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} \tilde{\boldsymbol{x}}_u^{(\ell)} \mathbb{1}_{u|v}. \qquad \mathbb{E}\left(L_{\text{batch}}\right) = \frac{1}{|\mathbb{G}|} \sum_{\mathcal{G}_s \in \mathbb{G}} \sum_{v \in \mathcal{V}_s} \frac{L_v}{\lambda_v} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} L_v.$$

from Prasanna et al., 2020

1. Use METIS for fast clustering estimation
2. Decompose stochastic blocks and connections between
3. Optimize block separately, drop intra-cluster connections

$$A = \bar{A} + \Delta = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}$$

$$\bar{A} = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix}$$



Algorithm 1: Cluster GCN

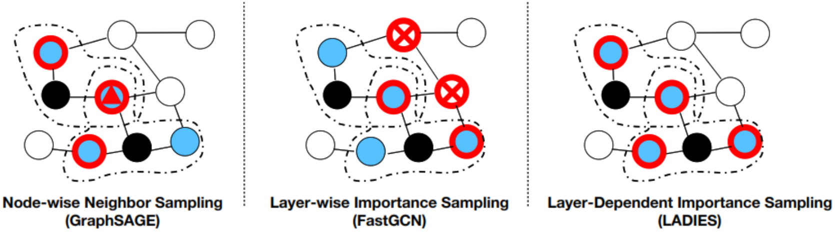
Input: Graph A, feature X, label Y;
Output: Node representation $\bar{X}$

Partition graph nodes into c clusters $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_c$ by METIS;
for $iter = 1, \cdots, max\_iter$ **do**
  Randomly choose q clusters, $t_1, \cdots, t_q$ from $\mathcal{V}$ without replacement;
  Form the subgraph $\bar{G}$ with nodes $\bar{\mathcal{V}} = [\mathcal{V}_{t_1}, \mathcal{V}_{t_2}, \cdots, \mathcal{V}_{t_q}]$ and links $A_{\bar{\mathcal{V}}, \bar{\mathcal{V}}}$;
  Compute $g \leftarrow \nabla \mathcal{L}_{A_{\bar{\mathcal{V}}, \bar{\mathcal{V}}}}$ (loss on the subgraph $A_{\bar{\mathcal{V}}, \bar{\mathcal{V}}}$) ;
  Conduct Adam update using gradient estimator g
Output: $\{W_l\}_{l=1}^{L}$

from Hsieh et al., 2019

LADIES: Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks

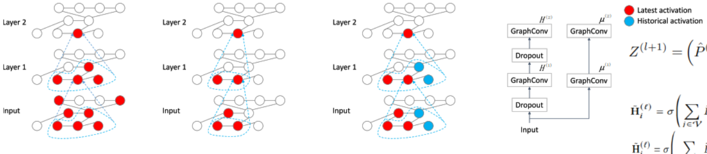
1. Node-wise neighbor-sampling method suffers from exponential growing neighbor size
2. Layer-wise importance-sampling method discards the neighbor-dependent constraints
3. LADIES selects neighborhood nodes, constructs a bipartite subgraph and computes the importance probability followed by sampling.



from Gu et al., 2019

VRGCN & MVS-GNN

1. Control layer-wise sampling to efficiently estimte next layer from historical embeddings
2. MVS-GNN uses one-shot sampling over nodes and batches with constant number of nodes



(a) Exact (b) Neighbour sampling (c) Control variate (d) CVD network

$$Z^{(l+1)} = \left(\hat{P}^{(l)}(H^{(l)} - \bar{H}^{(l)}) + P\bar{H}^{(l)}\right) W^{(l)}.$$

$$\bar{\mathbf{H}}_i^{(\ell)} = \sigma\left(\sum_{j\in\mathcal{V}} \tilde{L}_{i,j}^{(\ell)} \bar{\mathbf{H}}_i^{(\ell-1)} \mathbf{W}^{(\ell)}\right), \ \bar{\mathbf{H}}_i^{(\ell)} = \bar{\mathbf{H}}_i^{(\ell)}$$

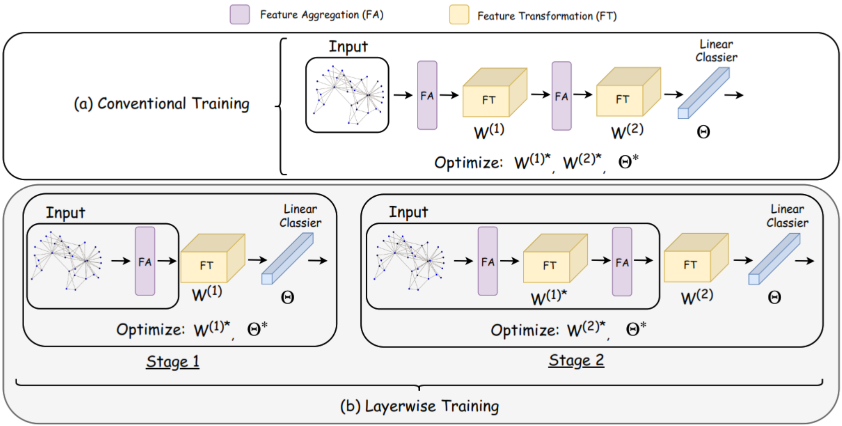
$$\tilde{\mathbf{H}}_i^{(\ell)} = \sigma\left(\sum_{j\in\mathcal{V}_{\ell-1}} \tilde{L}_{i,j}^{(\ell)} \tilde{\mathbf{H}}_i^{(\ell-1)} \mathbf{W}^{(\ell)} + \sum_{j\in\mathcal{V}\setminus\mathcal{V}_{\ell-1}} L_{i,j} \bar{\mathbf{H}}_i^{(\ell-1)} \mathbf{W}^{(\ell)}\right)$$

$$\bar{\mathbf{H}}_i^{(\ell)} = \tilde{\mathbf{H}}_i^{(\ell)}$$

from Song et al., 2018; Mahdavi et al., 2021

L^2–GCN: Layer-Wise and Learned Efficient Training of Graph Convolutional Networks

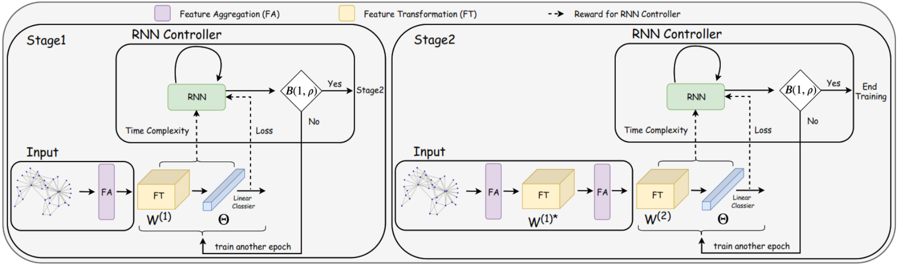
1. Layer-wise training



from Shen et al., 2020

L^2–GCN: Layer-Wise and Learned Efficient Training of Graph Convolutional Networks

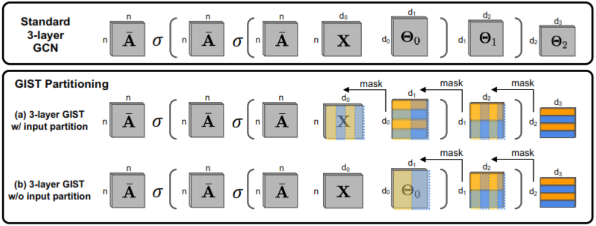
1. In each stage the layer-wise trained network generates the training loss,
2. The later index as the input for the RNN controller,
3. RNN controller outputs the hidden stage for the next epoch, and the stopping probability ρ for the current epoch

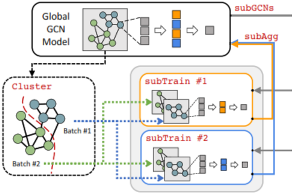


from Shen et al., 2020

GIST: Distributed Training for Large-Scale Graph Convolutional Networks

1. GCN partition into two sub-GCNs. Orange and blue colors depict different feature partitions.
2. subGCNs divides the global GCN into several sub-GCNs. Every subGCN is trained by subTrain using mini-batches (smaller sub-graphs) generated by Cluster.
3. SubGCN parameters are intermittently aggregated through subAgg.

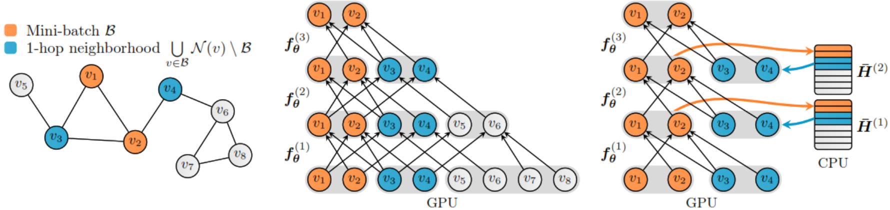




from Kyrillidis et al., 2021

GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings

1. Orange denotes the nodes in the current mini-batch and blue represents their 1-hop neighbors.
2. Historical embeddings avoid exponential growth of computations by pruning entire sub-trees of the computation graph



(a) Mini-batch selection (b) Original computation graph (c) GAS computation graph

from Leskovec et al., 2021

Comparison of Graphs Sampling models

TABLE XIV: Comprehensive summary of sampling methods in all categories

Category	Method	Random Sampling	Sampling Condition	Target Problem
Node-wise Sampling	GraphSAGE [35]	√	Random Neighbor Sampling	Inductive Learning
	PinSage [44]	×	Normalized Visit Counts	Scaling up GCN
	SSE [47]	√	Random Neighbor Sampling	Steady-state Condition Learning
	VR-GCN [48]	√	Random Neighbor Sampling	Receptive Field Reduction
Layer-wise Sampling	FastGCN [49]	×	Layer-independent Sampling	Neighbor Explosion Alleviation
	AS-GCN [50]	×	Layer-dependent Sampling	Neighbor Explosion Alleviation
	LADIES [51]	×	Layer-dependent Sampling	Sparse Connection Alleviation
Subgraph-based Sampling	Cluster-GCN [52]	√	Random Cluster Sampling	Constructing Graph Partition
	Parallelized Graph Sampling [55]	√	Parallel Frontier Sampling	Model parallelizing and Scaling up
	GraphSAINT [53]	×	Probabilistic Edge Sampling	Constructing Unbiased Subgraph Sampling
	RWT [54]	√	Random Neighbor Expansion	Handling Node Relation and Over-smoothing
Heterogeneous Sampling	Time-related Sampling [56]	×	Publish Time	Adversarial Action Alleviation
	HetGNN [57]	×	Visit Frequency	Heterogeneous Graph Learning
	HGSampling [58]	×	Probabilistic Sampling Nodes by Type	Heterogeneous Graph Learning
	Text Graph Sampling [59]	×	Intimacy Matrix	Heterogeneous Text Graph Learning

from Fan et al., 2021

References

- Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T. and Weinberger, K., 2019, May. Simplifying graph convolutional networks. In International conference on machine learning (pp. 6861-6871). PMLR.
- Zhou, K., Liu, N., Yang, F., Liu, Z., Chen, R., Li, L., Choi, S.H. and Hu, X., 2021. Adaptive Label Smoothing To Regularize Large-Scale Graph Training. arXiv preprint arXiv:2108.13555.
- Tailor, S.A., Opolka, F.L., Liò, P. and Lane, N.D., 2021. Adaptive Filters and Aggregator Fusion for Efficient Graph Convolutions. arXiv preprint arXiv:2104.01481.
- Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M. and Monti, F., 2020. Sign: Scalable inception graph neural networks. arXiv preprint arXiv:2004.11198.

References

- Li, G., Müller, M., Ghanem, B. and Koltun, V., 2021. Training Graph Neural Networks with 1000 Layers. arXiv preprint arXiv:2106.07476.
- Suresh, S., Budde, V., Neville, J., Li, P. and Ma, J., 2021. Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns. arXiv preprint arXiv:2106.06586.
- Zheng, D., Ma, C., Wang, M., Zhou, J., Su, Q., Song, X., Gan, Q., Zhang, Z. and Karypis, G., 2020, November. Distdgl: distributed graph neural network training for billion-scale graphs. In 2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3) (pp. 36-44). IEEE.
- Md, V., Misra, S., Ma, G., Mohanty, R., Georganas, E., Heinecke, A., Kalamkar, D., Ahmed, N.K. and Avancha, S., 2021. DistGNN: Scalable Distributed Training for Large-Scale Graph Neural Networks. arXiv preprint arXiv:2104.06700.

References

- Liu, X., Yan, M., Deng, L., Li, G., Ye, X. and Fan, D., 2021. Sampling methods for efficient training of graph convolutional networks: A survey. arXiv preprint arXiv:2103.05872.
- Karypis, G. and Kumar, V., 1997. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.
- Chiang, W.L., Liu, X., Si, S., Li, Y., Bengio, S. and Hsieh, C.J., 2019, July. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 257-266).
- Zeng, H., Zhou, H., Srivastava, A., Kannan, R. and Prasanna, V., 2019. Graphsaint: Graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931.

References

- You, Y., Chen, T., Wang, Z. and Shen, Y., 2020. L2-gcn: Layer-wise and learned efficient training of graph convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2127-2135).
- Bojchevski, A., Klicpera, J., Perozzi, B., Kapoor, A., Blais, M., Rózemberczki, B., Lukasik, M. and Günnemann, S., 2020, August. Scaling graph neural networks with approximate pagerank. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2464-2473).
- Wolfe, C.R., Yang, J., Chowdhury, A., Dun, C., Bayer, A., Segarra, S. and Kyrillidis, A., 2021. GIST: Distributed Training for Large-Scale Graph Convolutional Networks. arXiv preprint arXiv:2102.10424.
- Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y. and Gu, Q., 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. arXiv preprint arXiv:1911.07323.

References

- Cong, W., Forsati, R., Kandemir, M. and Mahdavi, M., 2020, August. Minimal variance sampling with provable guarantees for fast training of graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1393-1403).
- Chen, J., Zhu, J. and Song, L., 2017. Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.10568.
- Fey, M., Lenssen, J.E., Weichert, F. and Leskovec, J., 2021. GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings. arXiv preprint arXiv:2106.05609.
- Dai, H., Kozareva, Z., Dai, B., Smola, A. and Song, L., 2018, July. Learning steady-states of iterative algorithms over graphs. In International conference on machine learning (pp. 1106-1114). PMLR.