Scaling GNNs: Compression & Knowledge Distillation

I. Makarov & V. Pozdnyakov & D. Kiselev

BigData Academy MADE from Mail.ru Group

Graph Neural Networks and Applications

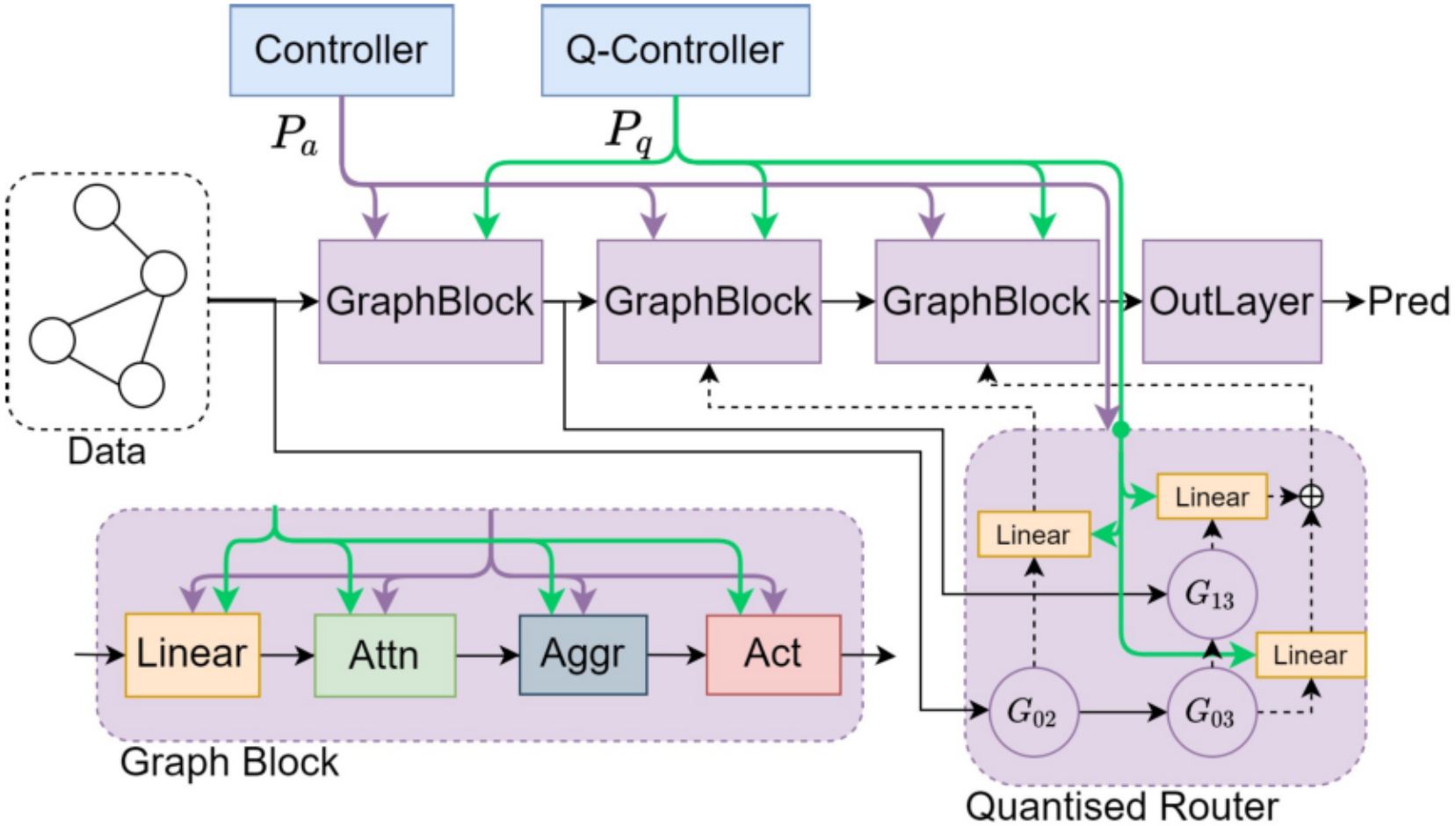
Topics

1. Quantization
2. Knowledge Distillation
3. Binarization
4. Federated Learning
5. Accelerating for Hardware

Quantization

Learned Low Precision Graph Neural Networks

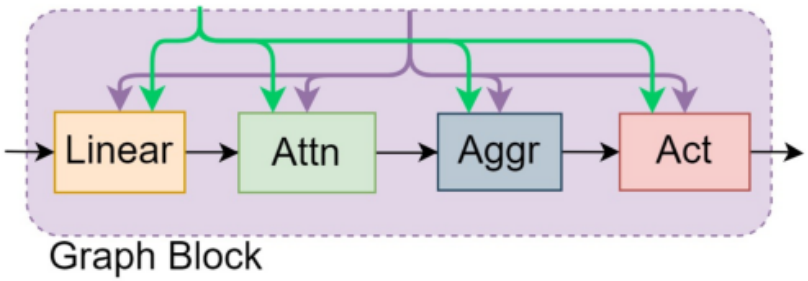
1. Use NAS for choosing proper aggregation, activation and precision layers



from Lio et al., 2020

Learned Low Precision Graph Neural Networks

1. Use NAS for choosing proper aggregation, activation and precision layers



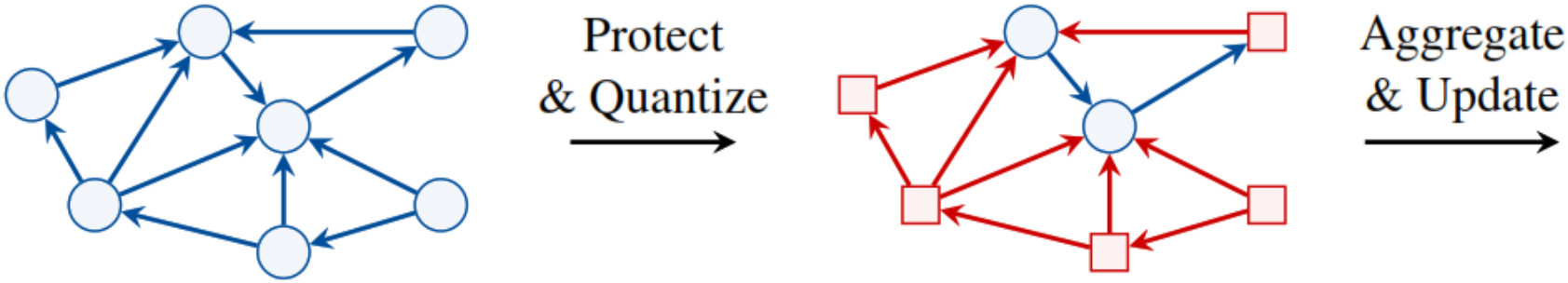
Attention Type	Equation
Const	$a_{ij} = 1$
GCN	$a_{ij} = \frac{1}{\sqrt{d_i d_j}}$
GAT	$a_{ij}^{gat} = \mathsf{LeakyReLU}(W_a(h_i \| \| h_j))$
Sym-GAT	$a_{ij} = a_{ij}^{gat} + a_{ji}^{gat}$
COS	$a_{ij} = < W_{a1}h_i, W_{a2}h_j >$
Linear	$a_{ij} = \tanh(\sum_{j \in N(i)}(W_a h_j))$
Gene-Linear	$a_{ij} = W_g \tanh(W_{a1}h_i + W_{a2}h_j)$

Activation	Equation
None	$f(x) = x$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Tanh	$f(x) = tanh(x)$
Softplus	$f(x) = \frac{1}{\beta}\log(1 + e^{\beta x})$
ReLU	$f(x) = Max(0, x)$
LeakyReLU	$f(x) = Max(0, x) + \alpha Min(0, x)$
ReLU6	$f(x) = Min(Max(0, x), 6)$
ELU	$f(x) = Max(0, x) + Min(0, \alpha(e^x - 1))$

from Lio et al., 2020

Quantization-aware training for graph neural networks

1. Retaining quality for high-degree nodes
2. Simplifying gradient update for low-degree nodes

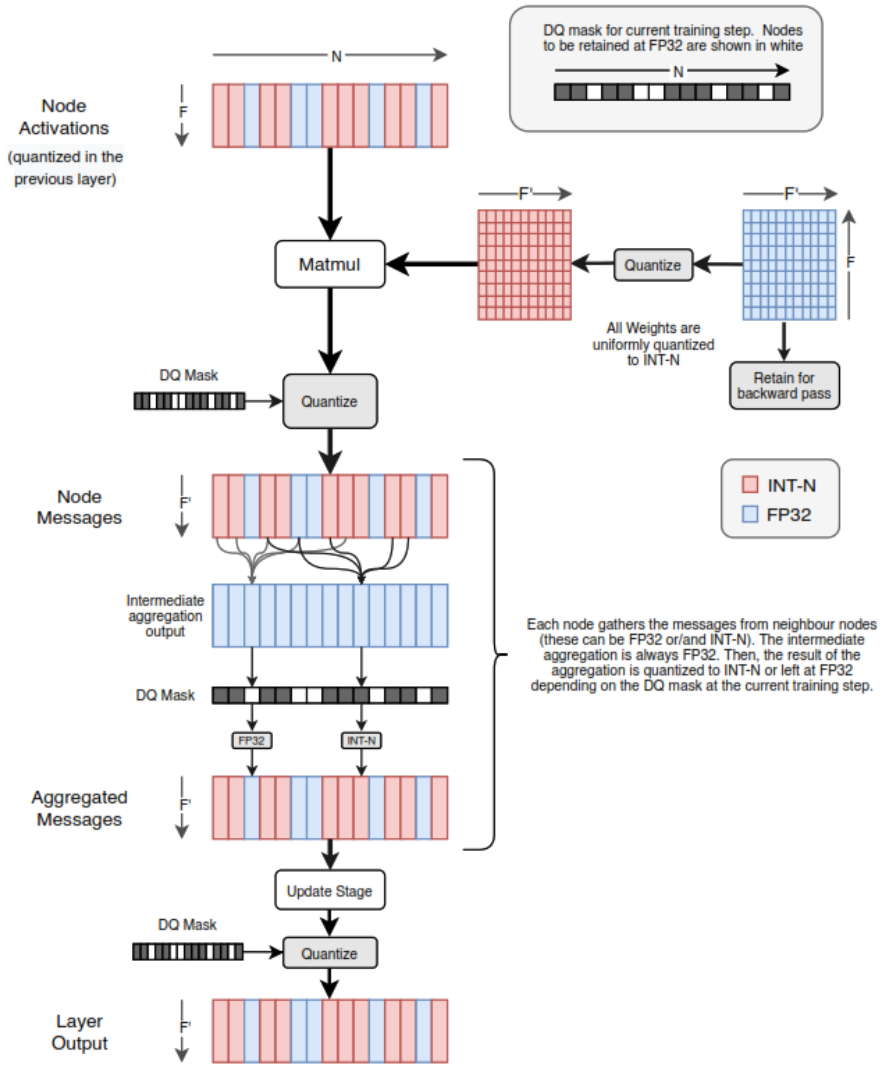


Algorithm 1 Degree-Quant (DQ). Functions accepting a protective mask $\mathbf{m}$ perform only the masked computations at full precision: intermediate tensors are *not* quantized. At test time protective masking is disabled. In fig. 11 (in the Appendix) we show with a diagram how a GCN layers makes use of DQ.

procedure TRAINFORWARDPASS($\mathcal{G}, \mathbf{p}$)
  ▷ Calculate mask and *quantized* weights, Θ', which all operations share
  $\mathbf{m} \leftarrow$ BERNOULLI($\mathbf{p}$)
  $\Theta' \leftarrow$ QUANTIZE(Θ)
  ▷ Messages with masked sources are at full precision (excluding weights)
  $\mathcal{M} \leftarrow$ MESSAGECALCULATE($\mathcal{G}, \Theta', \mathbf{m}$)
  $X \leftarrow$ QUANTIZE(AGGREGATE($\mathcal{M}, \Theta', \mathbf{m}$), $\mathbf{m}$) ▷ No quantization for masked nodes
  return UPDATE($X, \Theta', \mathbf{m}$) ▷ Quantized weights always used
end procedure

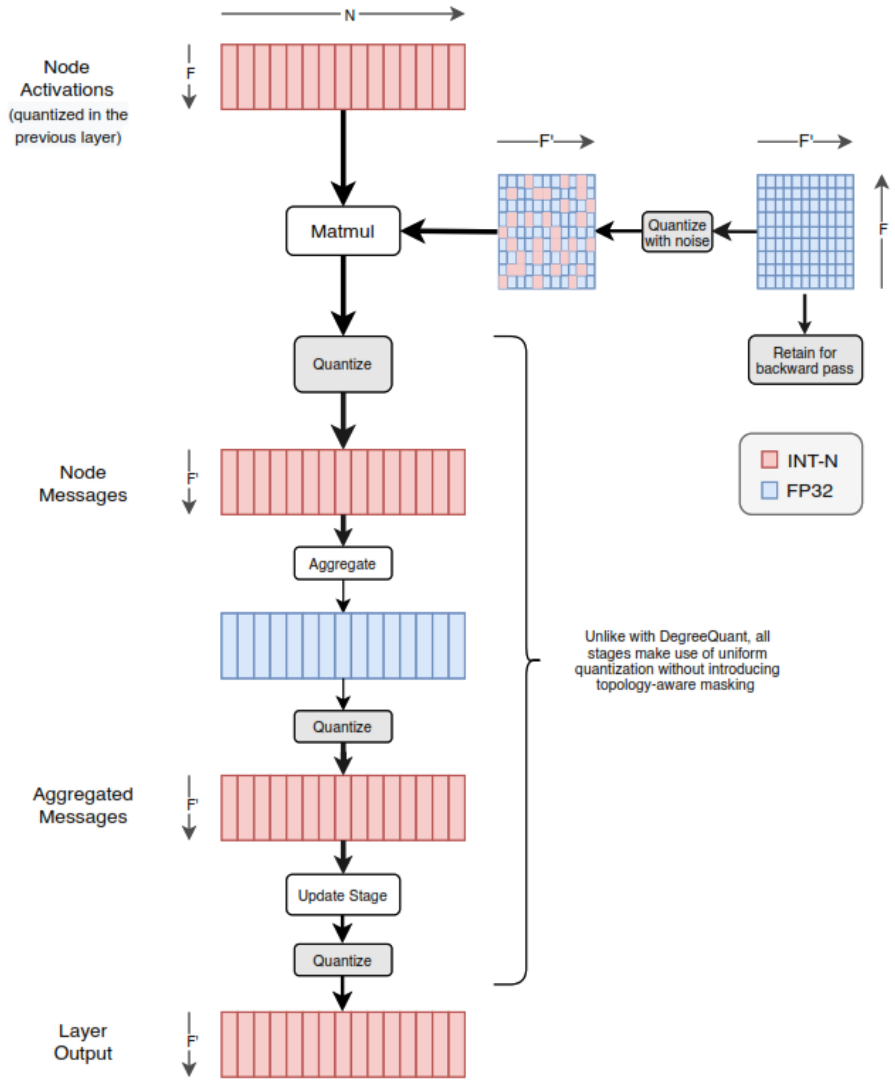
from Lane et al., 2021

Degree-based quantization



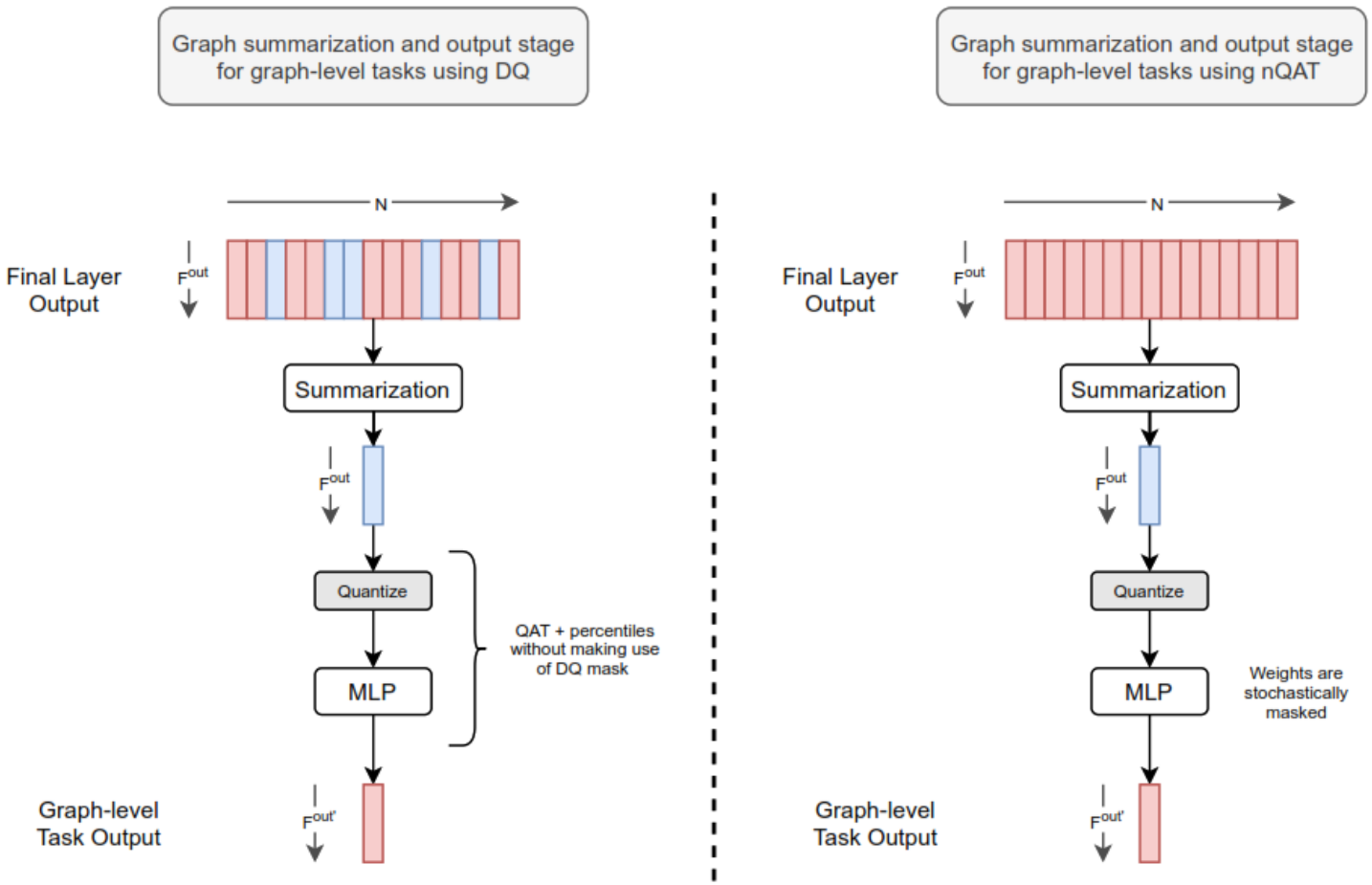
from Lane et al., 2021

Noise-based quantization



from Lane et al., 2021

Graph-level quantization

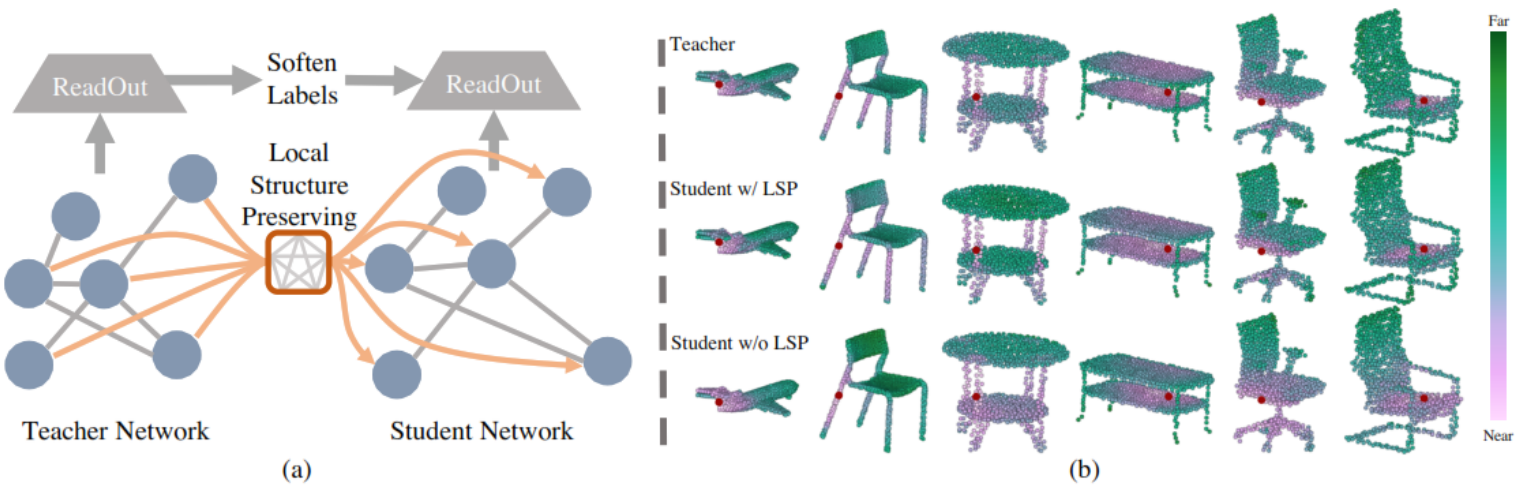


from Lane et al., 2021

Knowledge Distillation

Distilling Knowledge from Graph Convolutional Networks

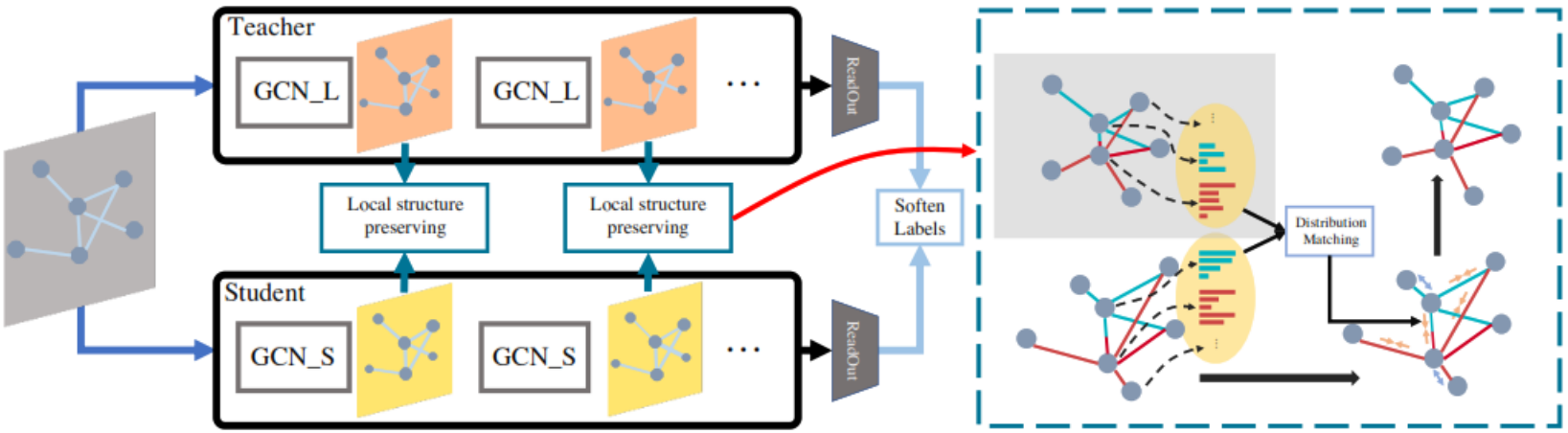
1. Distilling knowledge about how the teacher model embeds the topological structure
2. Structure of the feature space, visualized by the distance between the red point and the others on a point cloud dataset.
3. Top Row: structures obtained from the teacher;
4. Middle Row: structures obtained from the student trained with the local structure preserving (LSP) module;
5. Bottom Row: structures obtained from the student trained w/o LSP.



from Wang et al., 2021

Distilling Knowledge from Graph Convolutional Networks

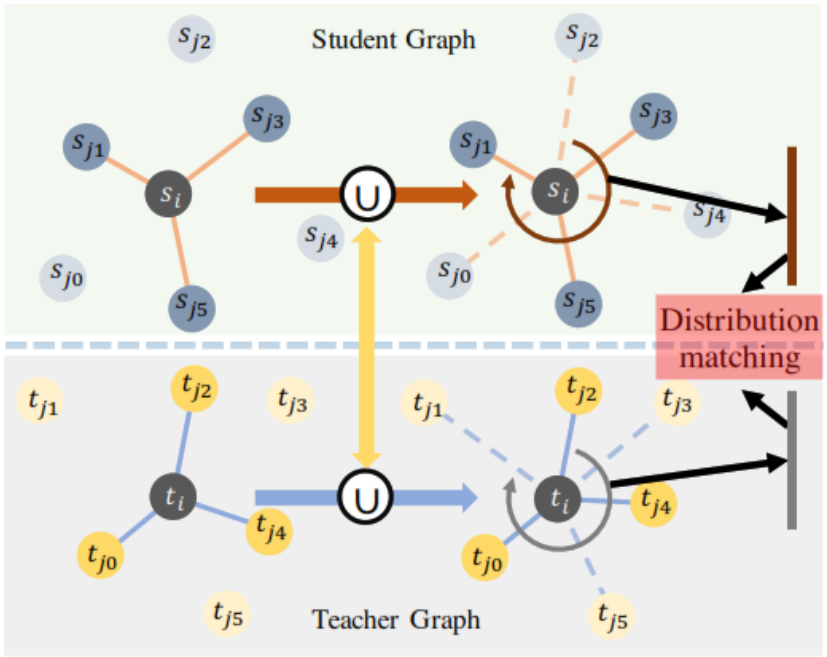
1. The local structure preserving module is the core of the proposed method.
2. Given the feature maps and the graphs from the teacher and the student, the distribution of the local structure for each node is matched between the distributions of the teacher and the student.



from Wang et al., 2021

Distilling Knowledge from Graph Convolutional Networks

1. Model learns structural similarity of teacher and student together with target cross-entropy.
2. To handle models with a dynamic graph, virtual edges are added according to the union of the two graphs from the student model and teacher model.



$$LS_{ij} = \frac{e^{\mathcal{SIM}(z_i, z_j)}}{\sum_{j:(j,i)\in\mathcal{E}} (e^{\mathcal{SIM}(z_i, z_j)})},$$

$$\mathcal{SIM}(z_i, z_j) = ||z_i - z_j||_2^2.$$

$$\mathcal{S}_i = D_{KL}(LS_i^s || LS_i^t) = \sum_{j:(j,i)\in\mathcal{E}} LS_{ij}^s log(\frac{LS_{ij}^s}{LS_{ij}^t})$$

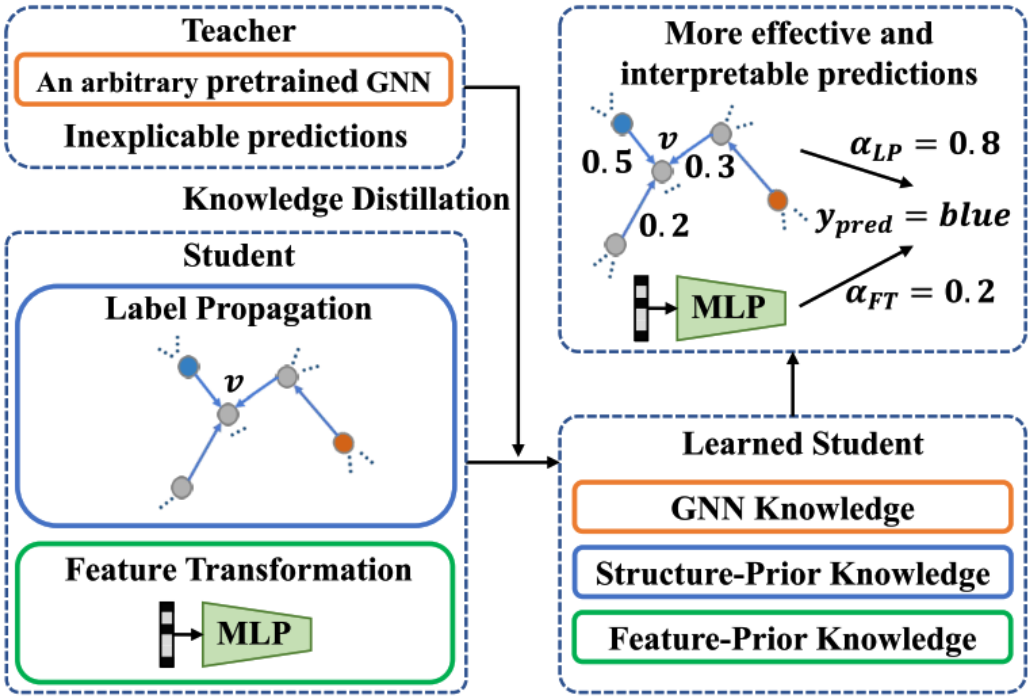
$$\mathcal{L}_{LSP} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{S}_i.$$

$$\mathcal{L} = \mathcal{H}(p_s, y) + \lambda \mathcal{L}_{LSP}$$

from Wang et al., 2021

Extract the Knowledge of Graph Neural Networks and Go Beyond it: An Effective Knowledge Distillation Framework

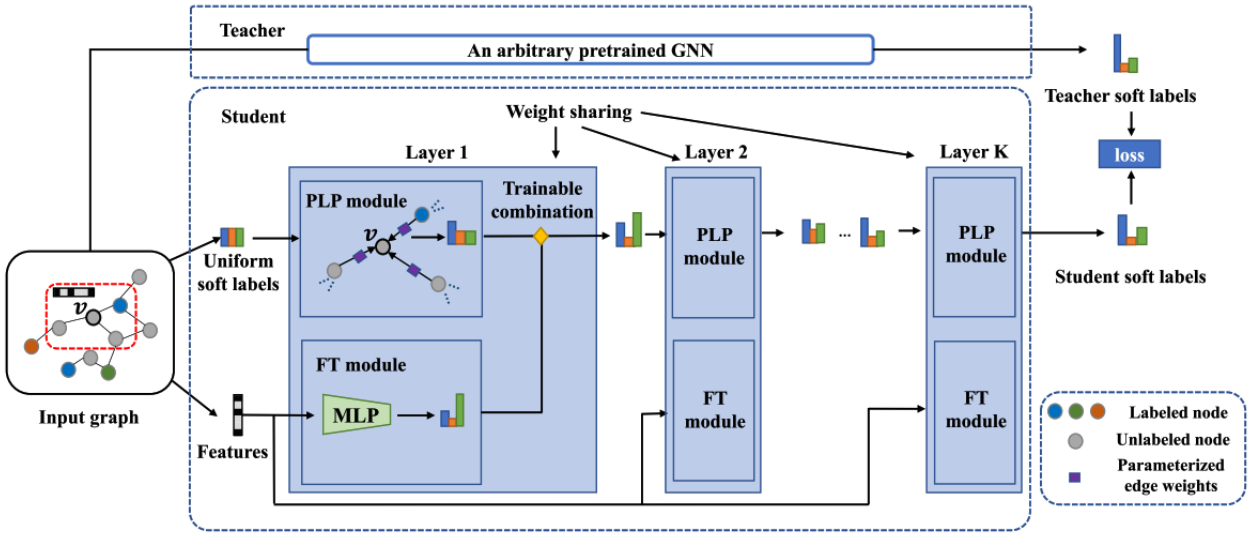
1. The knowledge in GNN teachers will be extracted and injected into the student during knowledge distillation.
2. The student can go beyond its corresponding teacher with more effective predictions.



from Shi et al., 2021

Extract the Knowledge of Graph Neural Networks and Go Beyond it: An Effective Knowledge Distillation Framework

1. Taking node v, the student model starts from node v's raw features and a uniform label distribution as soft labels.
2. Soft label prediction of v will be updated as a trainable combination of Parameterized Label Propagation (PLP) from v's neighbors and Feature Transformation (FT) of v's features.
3. Labels obtained by student and teacher will be matched.

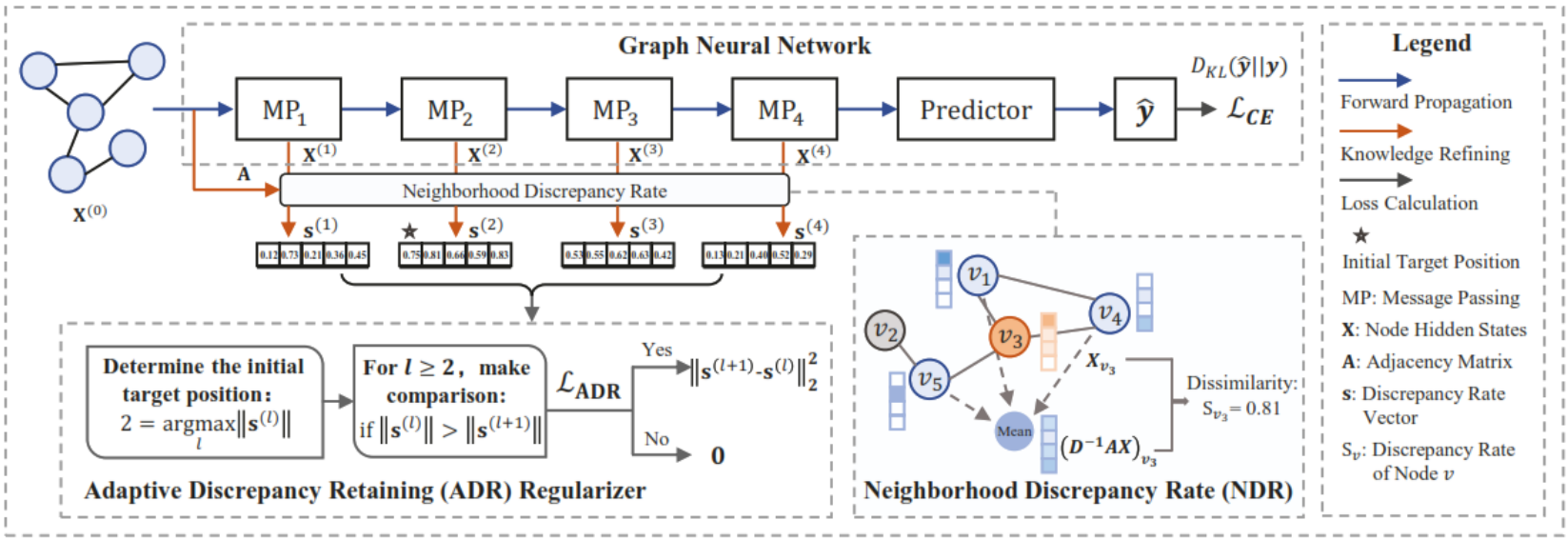


from Shi et al., 2021

Self-Distillation and SSL

On Self-Distilling Graph Neural Network

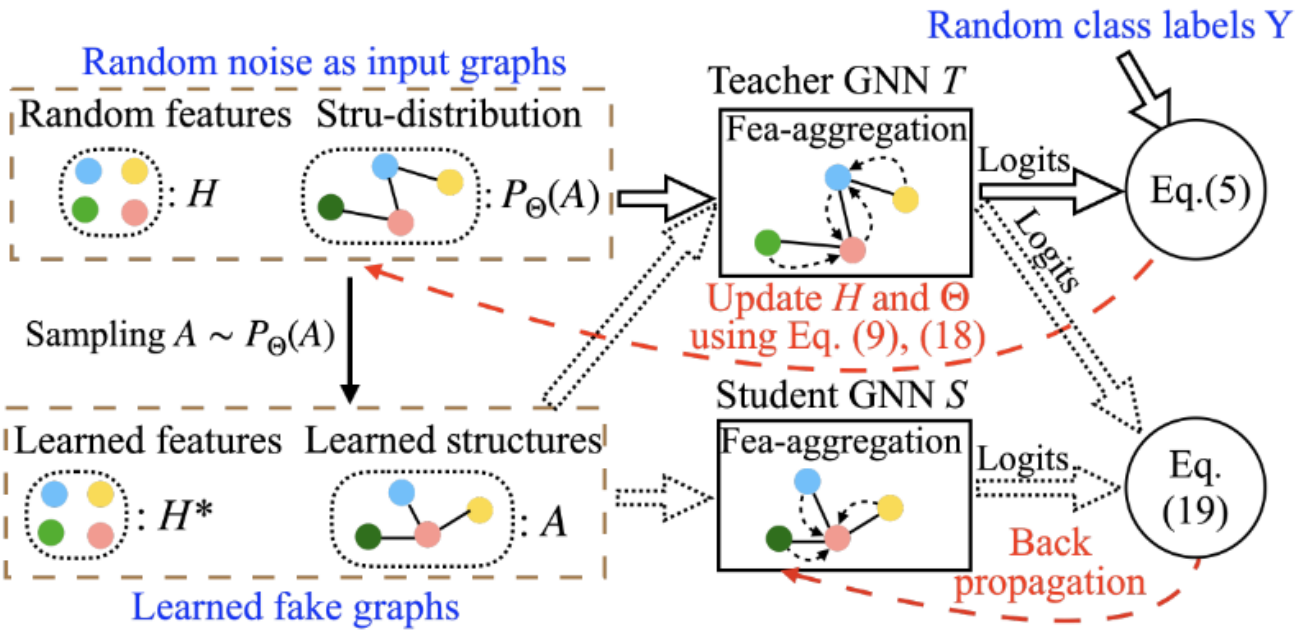
1. Neighborhood Discrepancy Rate to quantify the non-smoothness of each GNN layer via cosine distance of node embeddings and its neighborhood average embedding
2. Adaptive Discrepancy Retaining (ADR) for Stop Gradient regularization over discrepancy vectors over layers



from Huang et al., 2020

Graph-Free Knowledge Distillation for Graph Neural Networks

1. Distilling knowledge from a GNN with graph-free KD (GFKD)
2. GFKD learns graph topology structures for knowledge transfer by modeling them with multinomial distribution
3. Graph structures are obtained by only using GNN forward-propagation without back-propagation



from Zhang et al., 2021

Quantization by Distillation

Binary Graph Neural Networks

1. Binarization of message and aggregation
2. Distillation of proximity measure in Hamming space

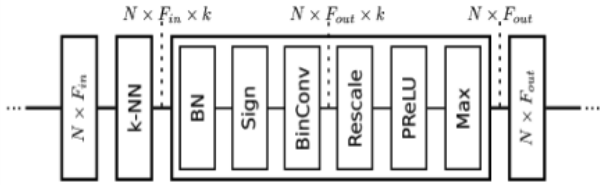
EdgeConv

$$\mathbf{x}_i^{(l)} = \gamma^{(l)}\left(\mathbf{x}_i^{(l-1)}, \square_{j\in\mathcal{N}(i)} \phi^{(l)}\left(\mathbf{x}_i^{(l-1)}, \mathbf{x}_j^{(l-1)}, \mathbf{e}_{ij}^{(l-1)}\right)\right)$$

$$\mathbf{e}_{ij}^{(l)} = \text{ReLU}\left(\boldsymbol{\theta}^{(l)}(\mathbf{x}_j^{(l-1)} - \mathbf{x}_i^{(l-1)}) + \boldsymbol{\phi}^{(l)}\mathbf{x}_i^{(l-1)}\right)$$
$$= \text{ReLU}\left(\boldsymbol{\Theta}^{(l)}\tilde{\mathbf{X}}^{(l-1)}\right)$$
$$\tilde{\mathbf{X}}^{(l-1)} = \left[\mathbf{x}_i^{(l-1)} \,||\, \mathbf{x}_j^{(l-1)} - \mathbf{x}_i^{(l-1)}\right]$$

Learnable Scaling < , >

$$\mathbf{A} \star \mathbf{B} \approx (\text{sign}(\mathbf{A}) \circledast \text{sign}(\mathbf{B})) \odot \Gamma$$



$$\mathbf{e}_{ij}^{(l)} = \sigma\left(\text{sign}(\boldsymbol{\Theta}^{(l)}) \circledast \text{sign}\left(\text{BN}\left(\tilde{\mathbf{X}}^{(l-1)}\right)\right) \odot \Gamma^{(l)}\right)$$

$$LS_{ij} = \frac{\exp(\text{SIM}(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{k\in\mathcal{N}(I)} \exp(\text{SIM}(\mathbf{x}_i, \mathbf{x}_k))}$$

$$L_{LSP} = \frac{1}{|\mathcal{V}|} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}^u(i)} LS_{ij}^s \log \frac{LS_{ij}^s}{LS_{ij}^t}$$

$$\text{SIM}(\mathbf{x}_i, \mathbf{x}_j) = e^{-||\mathbf{x}_i \oplus \mathbf{x}_j||_H}$$

$$\mathbf{x}_i^{(l)} = \text{Norm}\left(\text{ReLU}\left(\mathbf{W}^{(l)}\left[\mathbf{x}_i^{(l-1)} \,||\, \underset{j\in\mathcal{N}(i)}{\text{Aggr}}\, \mathbf{x}_j^{(l-1)}\right]\right)\right)$$

BinEdgeConv

$$\mathbf{h}^{(l)} = \text{sign}\left(\text{BN}\left(\left[\mathbf{x}_i^{(l-1)} \,||\, \underset{j\in\mathcal{N}(i)}{\text{Aggr}}\, \mathbf{x}_j^{(l-1)}\right]\right)\right)$$
$$\mathbf{x}_i^{(l)} = \text{Norm}\left(\sigma\left((\text{sign}(\mathbf{W}^{(l)}) \circledast \mathbf{h}^{(l)}) \odot \Gamma^{(l)}\right)\right)$$

XorEdgeConv

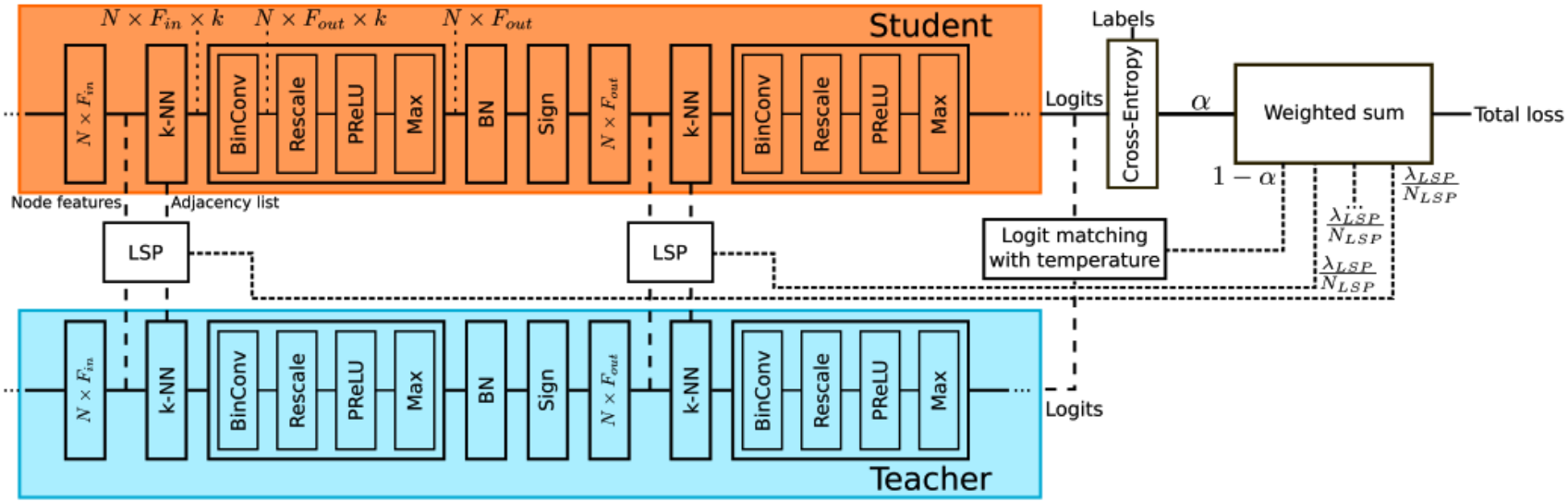
$$\mathbf{e}_{ij}^{(l)} = \sigma\left(\text{sign}(\boldsymbol{\Theta}^{(l)}) \circledast \tilde{\mathbf{X}}_b^{(l-1)} \odot \Gamma^{(l)}\right)$$
$$\mathbf{x}_i^{(l)} = \text{sign}\left(\max_{j\in\mathcal{N}(i)} \mathbf{e}_{ij}^{(l)}\right)$$
$$\tilde{\mathbf{X}}_b^{(l-1)} = \left[\mathbf{x}_i^{(l-1)} \,||\, -\mathbf{x}_j^{(l-1)} \odot \mathbf{x}_i^{(l-1)}\right]$$

$$\mathbf{x}_i^{(l)} = \max_{j\in\mathcal{N}(i)} \mathbf{e}_{ij}^{(l)} \qquad \mathbf{x}_i^{(l)} = \text{sign}\left(\text{BN}\left(\max_{j\in\mathcal{N}(i)} \mathbf{e}_{ij}^{(l)}\right)\right) \qquad \mathbf{x}_i^{(l)} = \text{sign}\left(\max_{j\in\mathcal{N}(i)} \text{BN}\left(\mathbf{e}_{ij}^{(l)}\right)\right)$$

from Zafeiriou et al., 2021

Binary Graph Neural Networks

1. Distillation with XorEdgeConv: the student model is more quantized than the teacher
2. Knowledge transfer points equipped with LSP modules encourage similar dynamic graph feature distributions after each k-NN graph computation
3. Logit matching is used to further inform the training of the student.

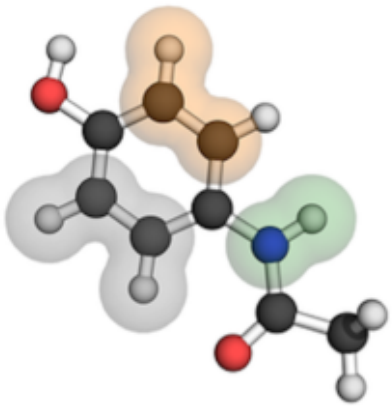


from Zafeiriou et al., 2021

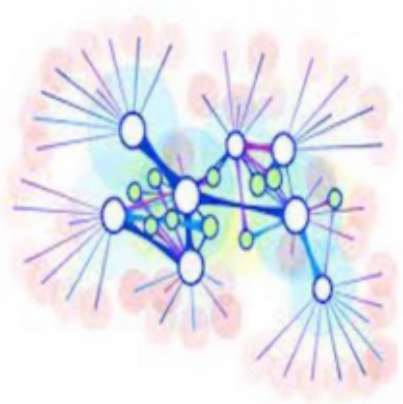
Federated Learning & Hardware Acceleration

FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks

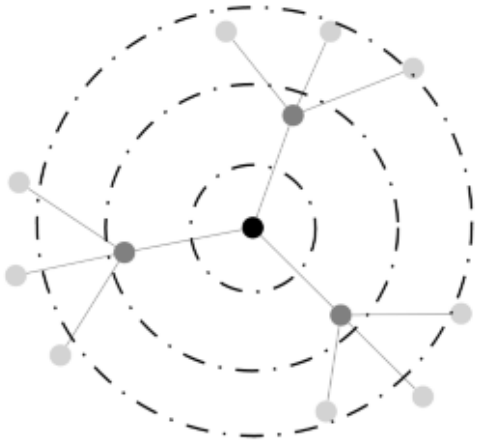
1. Federated Learning (FL) is a distributed learning paradigm that addresses data isolation problem
2. Data privacy and gradient-based data compromise lead to the necessity to adapt FL in various practical applications



(a) Graph-level FL



(b) Subgraph-level FL

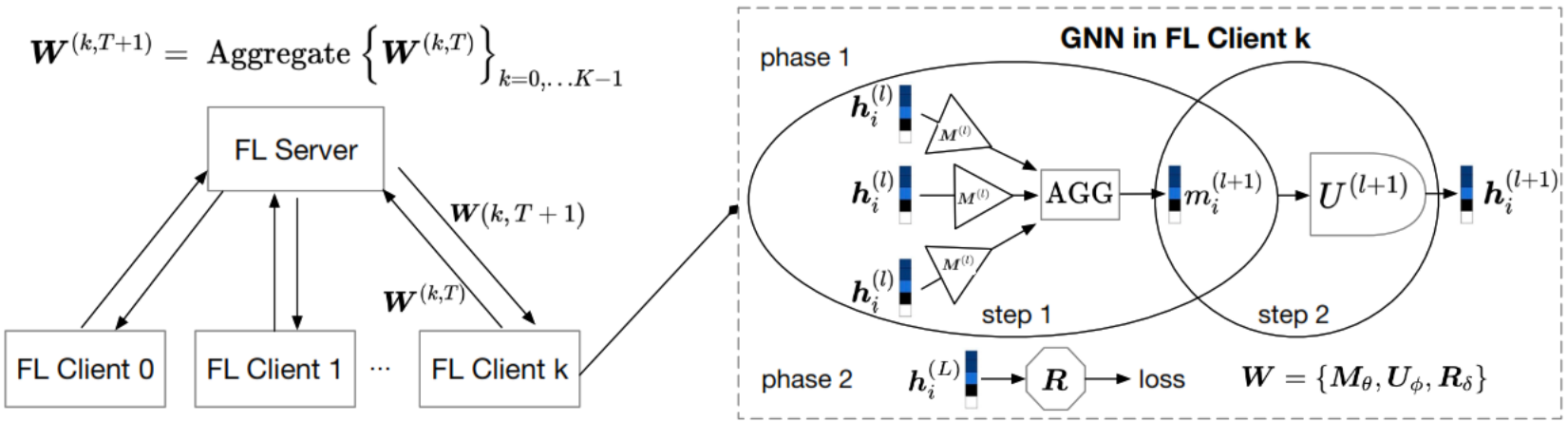


(c) Node-level FL

from Huang et al., 2021

FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks

1. No data/gradient/architecture based information is available, except for class/data volume distribution (possibly)
2. Simple averaging over weights is baseline idea similar to stochastic gradient descent



from Huang et al., 2021

Computing Graph Neural Networks

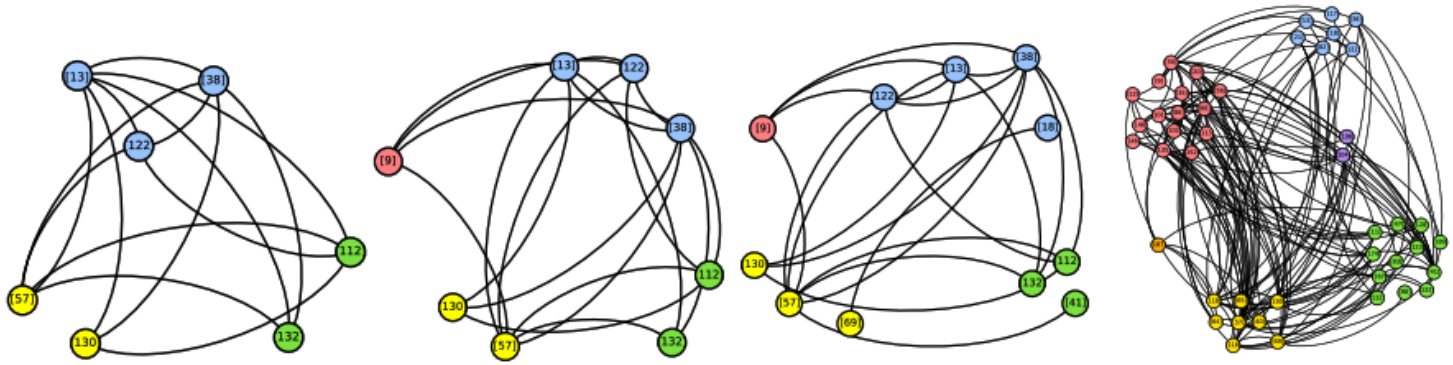
1. Different complexity, implementation and data processing aspects are of importance

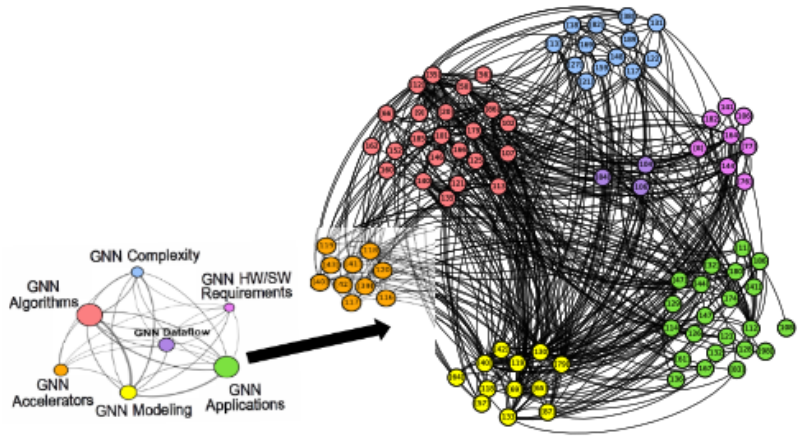
GNN Basics	GNN Evolution: State of the art	GNN Revolution: Accelerators	GNN Computing Vision
• Notation • General structure • Computing inference • Computing training	• A general perspective • An algorithm perspective	• Software frameworks and accelerators • Hardware accelerators	• SW-HW co-design • Graph awareness • Communication-centric design

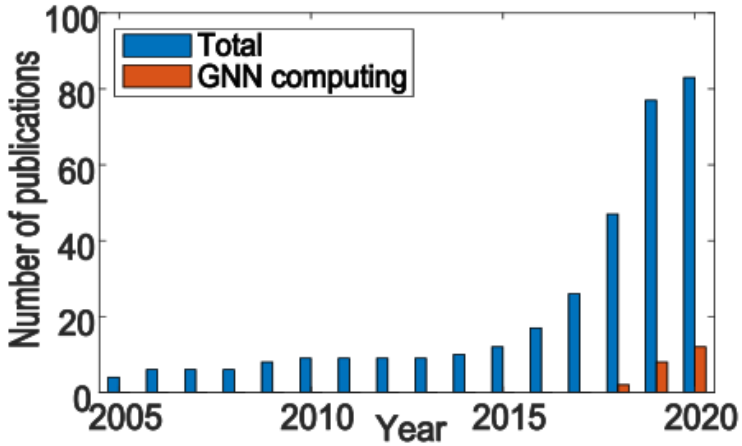
from Alarcon et al., 2021

Computing Graph Neural Networks

1. Evolution of Knowledge Graph for GNN implementations



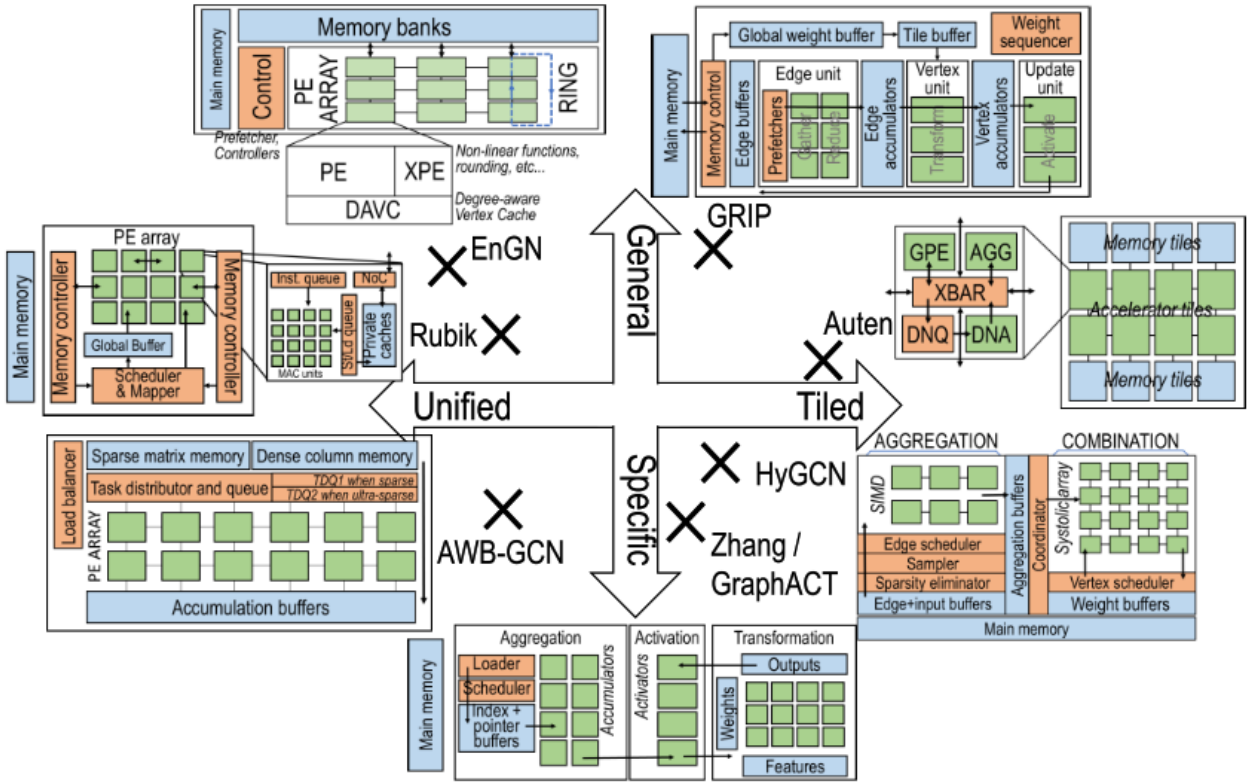




from Alarcon et al., 2021

Computing Graph Neural Networks

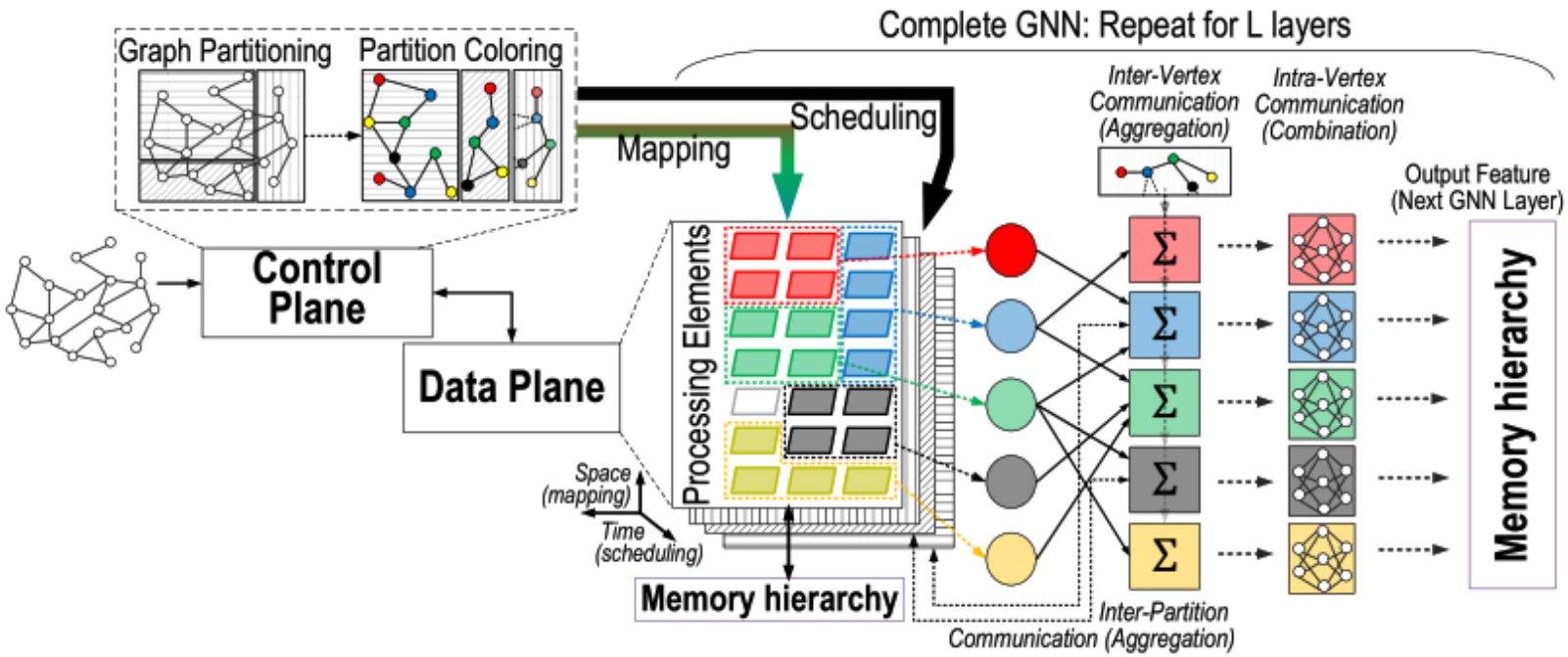
1. Qualitative classification and schematic representation of hardware accelerators for GNN inference
2. Green, blue, and red squares represent processors, memory, and control units, respectively.



from Alarcon et al., 2021

Computing Graph Neural Networks

1. Architectural vision for GNN accelerators with hardware-software co-design (control and data planes), graph awareness (guided mapping and scheduling), and communication-centric design (reconfigurable interconnect)



from Alarcon et al., 2021

References

- Zhao, Y., Wang, D., Bates, D., Mullins, R., Jamnik, M. and Lio, P., 2020. Learned Low Precision Graph Neural Networks. arXiv preprint arXiv:2009.09232.
- Tailor, S.A., Fernandez-Marques, J. and Lane, N.D., 2020. Degree-quant: Quantization-aware training for graph neural networks. arXiv preprint arXiv:2008.05000.
- Bahri, M., Bahl, G. and Zafeiriou, S., 2021. Binary Graph Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9492-9501).
- Yang, Y., Qiu, J., Song, M., Tao, D. and Wang, X., 2020. Distilling knowledge from graph convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7074-7083).

References

- Yang, C., Liu, J. and Shi, C., 2021, April. Extract the Knowledge of Graph Neural Networks and Go Beyond it: An Effective Knowledge Distillation Framework. In Proceedings of the Web Conference 2021 (pp. 1227-1237).
- Chen, Y., Bian, Y., Xiao, X., Rong, Y., Xu, T. and Huang, J., 2020. On Self-Distilling Graph Neural Network. arXiv preprint arXiv:2011.02255.
- Deng, X. and Zhang, Z., 2021. Graph-Free Knowledge Distillation for Graph Neural Networks. arXiv preprint arXiv:2105.07519.
- He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., He, L., Yang, L., Philip, S.Y., Rong, Y. and Zhao, P., 2021. FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks.
- Abadal, S., Jain, A., Guirado, R., López-Alonso, J. and Alarcón, E., 2021. Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys (CSUR), 54(9), pp.1-38.