DEGREE-QUANT: QUANTIZATION-AWARE TRAINING FOR GRAPH NEURAL NETWORKS

Shyam A. Tailor*
Department of Computer Science & Technology
University of Cambridge

Javier Fernandez-Marques*
Department of Computer Science
University of Oxford

Nicholas D. Lane
Department of Computer Science and Technology
University of Cambridge
& Samsung AI Center

Background: Graph Neural Networks

- GNNs are built to model irregularly structured data
- Recent advancements have centered around:
  - More sophisticated models: GCN, GAT, GIN…
  - Better aggregation function (Corso et al., 2020)
    - Specialized neighborhood aggregation for graphs

Background: Message Passing Neural Networks

- Many GNN architectures can be modeled in MPNN paradigm

- Message Passing Neural Networks (MPNN)
  adopt by current frameworks, such as *PyTorch Geometrics* (Matthias Fey et al., 2019) and *Deep Graph Learning* (Minjie Wang et al., 2020)

1. Gather and transform the messages from neighbors

$$\mathbf{m_i^{l+1}} = AGG(\{\phi^{\mathbf{l+1}}(\mathbf{h_i^l}, \mathbf{h_j^l}, \mathbf{e_{ij}}) \mid j \in N(i)\})$$

2. Update the state of the target node

$$\mathbf{h_i^{l+1}} = \gamma^{l+1}(\mathbf{h_i^l}, \mathbf{m_i^{l+1}})$$

Graph Neural Network Efficiency

- There remains little works addressing GNN efficiency!
- Computation characteristics:
  - GNN have few model parameters (<1MB mostly)
  - The computation remains tightly coupled with input graph size

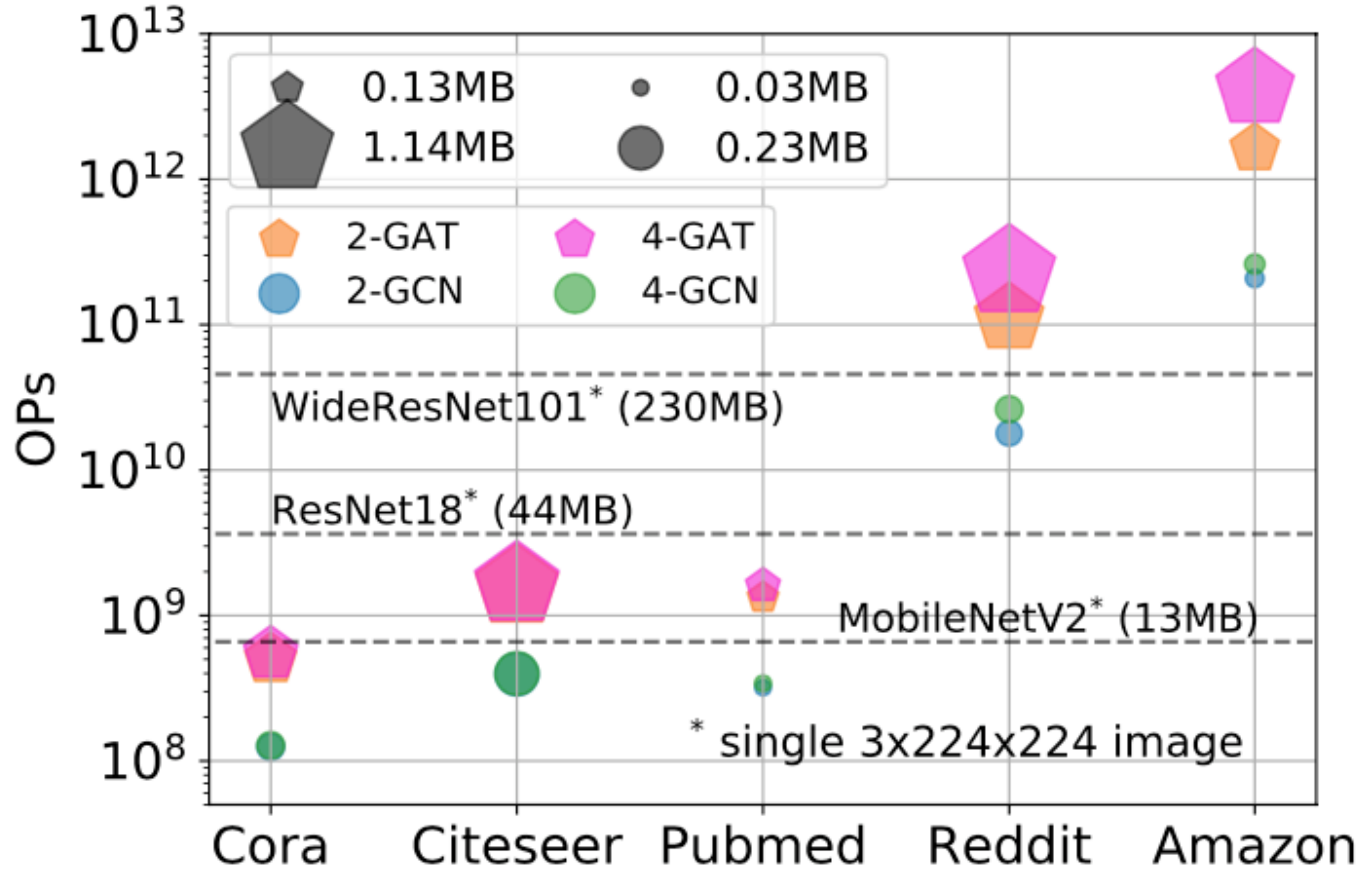

Figure 1: Despite GNN model sizes rarely exceeding 1MB, the OPs needed for inference grows at least linearly with the size of the dataset and node features. GNNs with models sizes 100× smaller than popular CNNs require many more OPs to process large graphs.

Graph Neural Network Efficiency (Cont.)

- Challenge: enable GNNs perform inference efficiently
  - GNNs have been combined with CNNs for SLAM feature matching (Sarlin et al., 2019)
- Integer quantization is one way to lower the computation, especially for the mobile devices
  - No work has addressed this issue: quantizing GNNs and showing latency benefit

Quantization

- Quantization-aware training (QAT) has become the de-facto approach
- QAT schemes involve exposing the numerical errors
  - simulating it on the forward pass

$$x_q = \min(q_{\max}, \max(q_{\min}, \lfloor x/s + z \rceil))$$

  - make use of Straight-Through Estimation (STE) to compute the gradients

Source of Error: STE

- The choice of STE implementation generally results in marginal difference for CNNs
- But the implementation will have a large impact on GNNs

Dataset	Model Arch.	*vanilla* STE				STE with Gradient Clipping			
		min/max		momentum		min/max		momentum	
		W8A8	W4A4	W8A8	W4A4	W8A8	W4A4	W8A8	W4A4
Cora (Acc. %) $\uparrow$	GCN	**81.0 ± 0.7**	65.3 ± 4.9	42.3 ± 11.1	49.4 ± 8.8	80.8 ± 0.8	62.3 ± 5.2	66.9 ± 18.2	**77.2 ± 2.5**
	GAT	76.0 ± 2.2	16.8 ± 8.5	81.7 ± 1.3	**51.7 ± 5.8**	76.4 ± 2.6	15.4 ± 8.1	**81.9 ± 0.7**	47.4 ± 5.0
	GIN	69.9 ± 1.9	25.9 ± 2.6	49.2 ± 10.2	**42.8 ± 4.0**	69.2 ± 2.3	29.5 ± 3.5	**75.1 ± 1.1**	40.5 ± 5.0
MNIST (Acc. %) $\uparrow$	GCN	**90.4 ± 0.2**	51.3 ± 7.5	90.1 ± 0.5	**70.6 ± 2.4**	90.4 ± 0.3	54.8 ± 1.5	90.2 ± 0.4	10.3 ± 0.0
	GAT	**95.8 ± 0.1**	20.1 ± 3.3	95.7 ± 0.3	67.4 ± 3.2	95.7 ± 0.1	30.2 ± 7.4	95.7 ± 0.3	**76.3 ± 1.2**
	GIN	96.5 ± 0.3	62.4 ± 21.8	**96.7 ± 0.2**	**91.0 ± 0.6**	96.4 ± 0.4	19.5 ± 2.1	75.3 ± 18.1	10.8 ± 0.9
ZINC (Loss) $\downarrow$	GCN	0.486 ± 0.01	0.747 ± 0.02	0.509 ± 0.01	0.710 ± 0.05	0.495 ± 0.01	0.766 ± 0.02	**0.483 ± 0.01**	**0.692 ± 0.01**
	GAT	0.471 ± 0.01	0.740 ± 0.02	0.571 ± 0.03	**0.692 ± 0.06**	0.466 ± 0.01	0.759 ± 0.04	**0.463 ± 0.01**	0.717 ± 0.03
	GIN	0.393 ± 0.02	1.206 ± 0.27	**0.386 ± 0.03**	**0.572 ± 0.02**	0.390 ± 0.02	1.669 ± 0.10	0.388 ± 0.02	0.973 ± 0.24

Table 1: Impact on performance of four typical quantization implementations for INT8 and INT4. The configuration that resulted in best performing models for each dataset-model pair is bolded. Hyperparameters for each experiment were fine-tuned independently. As expected, adding clipping does not change performance with min/max but does with momentum. **A major contribution of this work is identifying that seemingly unimportant choices in quantization implementation cause dramatic changes in performance.**

1. All INT4 experiments benefit from momentum
2. GIN models often suffer from higher performance degradation

Source of Error: Node Degree

- Aggregation phase introduce substantial numerical errors
  - As degree increases, the variance of aggregation values will increase
- Suppose incoming message values are X_i, aggregation output is Y_n, n is the number of degree
  - For GIN layer: $\mathbb{E}[Y_n] = O(n)$, $VAR[Y_n] = O(n)$
  - For GCN layer: $\mathbb{E}[Y_n] = O(\sqrt{n})$
  - For GAT layer: $\mathbb{E}[Y_n] = O(1)$

Source of Error: Node Degree (Cont.)

- Empirical validation

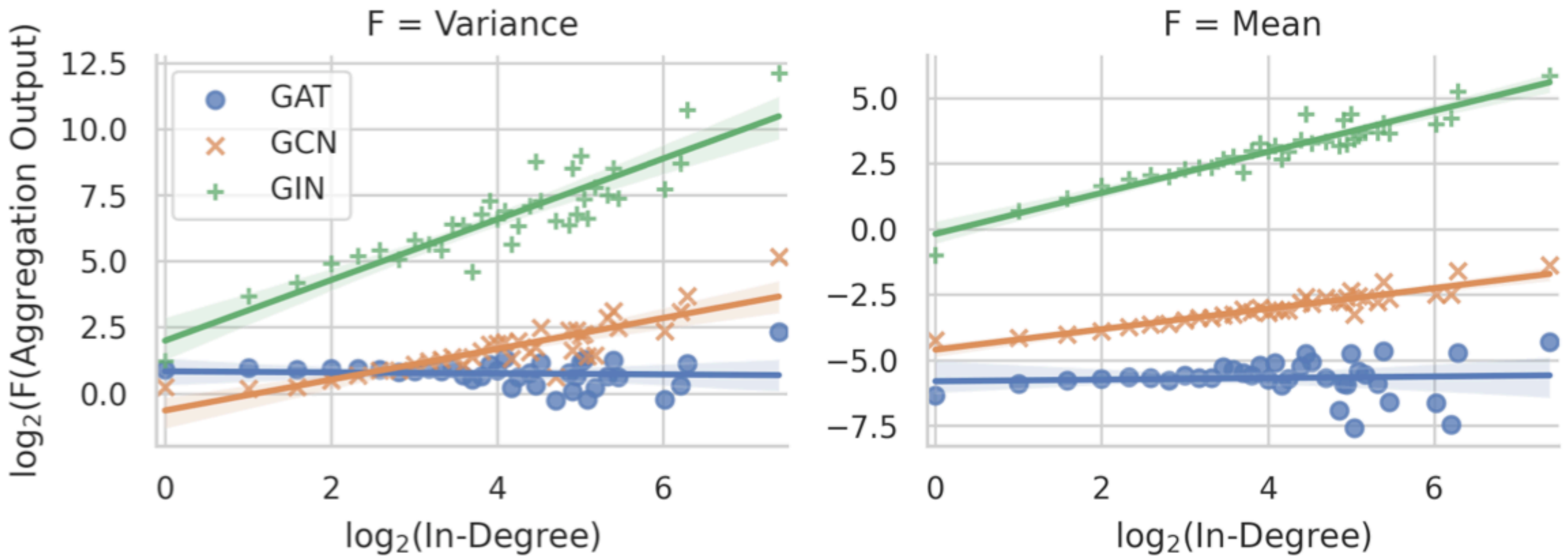

Figure 3: Analysis of values collected immediately after aggregation at the final layer of FP32 GNNs trained on Cora. Generated using channel data collected from 100 runs for each architecture. As in-degree grows, so does the mean and variance of channel values after aggregation.

- Gradients are also incorrect for the weights

GIN

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{i=1}^{|V|} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{l+1}^{(i)}} \circ f'(\mathbf{W}\mathbf{y}_{\mathrm{GIN}}^{(i)}) \right) \mathbf{y}_{\mathrm{GIN}}^{(i)\top}$$

GCN

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{l+1}^{(i)}} \circ f'(\mathbf{y}_{\mathrm{GCN}}^{(i)}) \right) \mathbf{h}_l^{(j)\top}$$

Degree-Quant

- Stochastic Protection: fix incorrect weight updates
    - Protective node mask is generated; masked nodes perform full-precision
    - Mask is generated by Bernoulli distribution: probability is of proportion to node degree

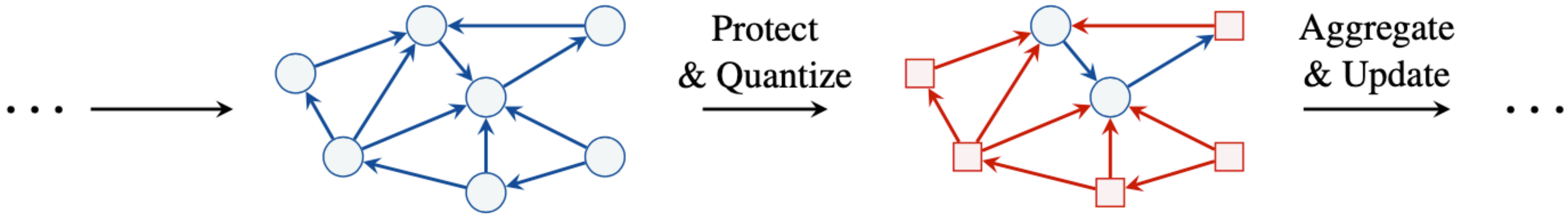

Figure 4: High-level view of the stochastic element of Degree-Quant. Protected (high in-degree) nodes, in blue, operate at full precision, while unprotected nodes (red) operate at reduced precision. High in-degree nodes contribute most to poor gradient estimates, hence they are stochastically protected from quantization more often.

Degree-Quant (Conti.)

- Percentile tracking of Quantization Ranges
    - Order the values in the tensor
    - Clip a fraction of the values at both ends of the distribution
    - Quantization ranges are more representative of the majority of values

Experimental Results: Accuracy

- GIN is less resilient to quantization
- nQAT helps on citation datasets (but not on others)
- DQ is comparable to FP32 when quantized to 8-bits
- DQ improves a lot in 4-bit quantization

Quant. Scheme	Model Arch.	Node Classification (Accuracy %) Cora ↑	Citeseer ↑	Graph Classification (Accuracy %) MNIST ↑	CIFAR-10 ↑	Graph Regression (Loss) ZINC ↓
Ref. (FP32)	GCN	81.4 ± 0.7	71.1 ± 0.7	90.0 ± 0.2	54.5 ± 0.1	0.469 ± 0.002
	GAT	83.1 ± 0.4	72.5 ± 0.7	95.6 ± 0.1	65.4 ± 0.4	0.463 ± 0.002
	GIN	77.6 ± 1.1	66.1 ± 0.9	93.9 ± 0.6	53.3 ± 3.7	0.414 ± 0.009
Ours (FP32)	GCN	81.2 ± 0.6	71.4 ± 0.9	90.9 ± 0.4	58.4 ± 0.5	0.450 ± 0.008
	GAT	83.2 ± 0.3	72.4 ± 0.8	95.8 ± 0.4	65.1 ± 0.8	0.455 ± 0.006
	GIN	77.9 ± 1.1	65.8 ± 1.5	96.4 ± 0.4	57.4 ± 0.7	0.334 ± 0.024
QAT (W8A8)	GCN	81.0 ± 0.7	71.3 ± 1.0	90.9 ± 0.2	56.4 ± 0.5	0.481 ± 0.029
	GAT	81.9 ± 0.7	71.2 ± 1.0	95.8 ± 0.3	66.3 ± 0.4	0.460 ± 0.005
	GIN	75.6 ± 1.2	63.0 ± 2.6	96.7 ± 0.2	52.4 ± 1.2	0.386 ± 0.025
nQAT (W8A8)	GCN	81.0 ± 0.8	70.7 ± 0.8	91.1 ± 0.1	56.2 ± 0.5	0.472 ± 0.015
	GAT	82.5 ± 0.5	71.2 ± 0.7	96.0 ± 0.1	66.7 ± 0.2	0.459 ± 0.007
	GIN	77.4 ± 1.3	65.1 ± 1.4	96.4 ± 0.3	52.7 ± 1.4	0.405 ± 0.016
DQ (W8A8)	GCN	81.7 ± 0.7 (+**0.7**)	71.0 ± 0.9 (-**0.3**)	90.9 ± 0.2 (-**0.2**)	56.3 ± 0.1 (-**0.1**)	0.434 ± 0.009 (+**9.8**)
	GAT	82.7 ± 0.7 (+**0.2**)	71.6 ± 1.0 (+**0.4**)	95.8 ± 0.4 (-**0.2**)	67.7 ± 0.5 (+**1.0**)	0.456 ± 0.005 (+**0.9**)
	GIN	78.7 ± 1.4 (+**1.3**)	67.5 ± 1.4 (+**2.4**)	96.6 ± 0.1 (-**0.1**)	55.5 ± 0.6 (+**2.8**)	0.357 ± 0.014 (+**7.5**)
QAT (W4A4)	GCN	77.2 ± 2.5	64.1 ± 4.1	70.6 ± 2.4	38.1 ± 1.6	0.692 ± 0.013
	GAT	55.6 ± 5.4	65.3 ± 1.9	76.3 ± 1.2	41.0 ± 1.1	0.655 ± 0.032
	GIN	42.5 ± 4.5	18.6 ± 2.9	91.0 ± 0.6	45.6 ± 3.6	0.572 ± 0.02
nQAT (W4A4)	GCN	78.1 ± 1.5	65.8 ± 2.6	70.9 ± 1.5	40.1 ± 0.7	0.669 ± 0.128
	GAT	54.9 ± 5.6	65.5 ± 1.7	78.4 ± 1.5	41.0 ± 0.6	0.637 ± 0.012
	GIN	45.0 ± 5.0	34.6 ± 3.8	91.3 ± 0.5	48.7 ± 1.7	0.561 ± 0.068
DQ (W4A4)	GCN	78.3 ± 1.7 (+**0.2**)	66.9 ± 2.4 (+**1.1**)	84.4 ± 1.3 (+**13.5**)	51.1 ± 0.7 (+**11.0**)	0.536 ± 0.011 (+**26.2**)
	GAT	71.2 ± 2.9 (+**16.3**)	67.6 ± 1.5 (+**2.1**)	93.1 ± 0.3 (+**14.7**)	56.5 ± 0.6 (+**15.5**)	0.520 ± 0.021 (+**20.6**)
	GIN	69.9 ± 3.4 (+**24.9**)	60.8 ± 2.1 (+**26.2**)	95.5 ± 0.4 (+**4.2**)	50.7 ± 1.6 (+**2.0**)	0.431 ± 0.012 (+**23.2**)

Experimental Results: latency

- INT-8 algorithmic can accelerate inference up to **4.7x**
- GPU has less benefit due to their massively-parallel nature

Device	Arch.	Zinc (Batch=10K) FP32	Zinc (Batch=10K) W8A8	Zinc (Batch=10K) Speedup	Reddit FP32	Reddit W8A8	Reddit Speedup
CPU	GCN	181ms	42ms	4.3×	13.1s	3.1s	4.2×
	GAT	190ms	50ms	3.8×	13.1s	2.8s	4.7×
	GIN	182ms	43ms	4.2×	13.1s	3.1s	4.2×
GPU	GCN	39ms	31ms	1.3×	191ms	176ms	1.1×
	GAT	17ms	15ms	1.1×	OOM	OOM	-
	GIN	39ms	31ms	1.3×	191ms	176ms	1.1×

Table 4: INT8 latency results run on a 22 core 2.1GHz Intel Xeon Gold 6152 and, on a GTX 1080Ti GPU. Quantization provides large speedups on a variety of graphs for CPU and non-negligible speedups with unoptimized INT8 GPU kernels.

Ablation Studies

- Benefits of percentile Ranges
- Min-Max doubles the range

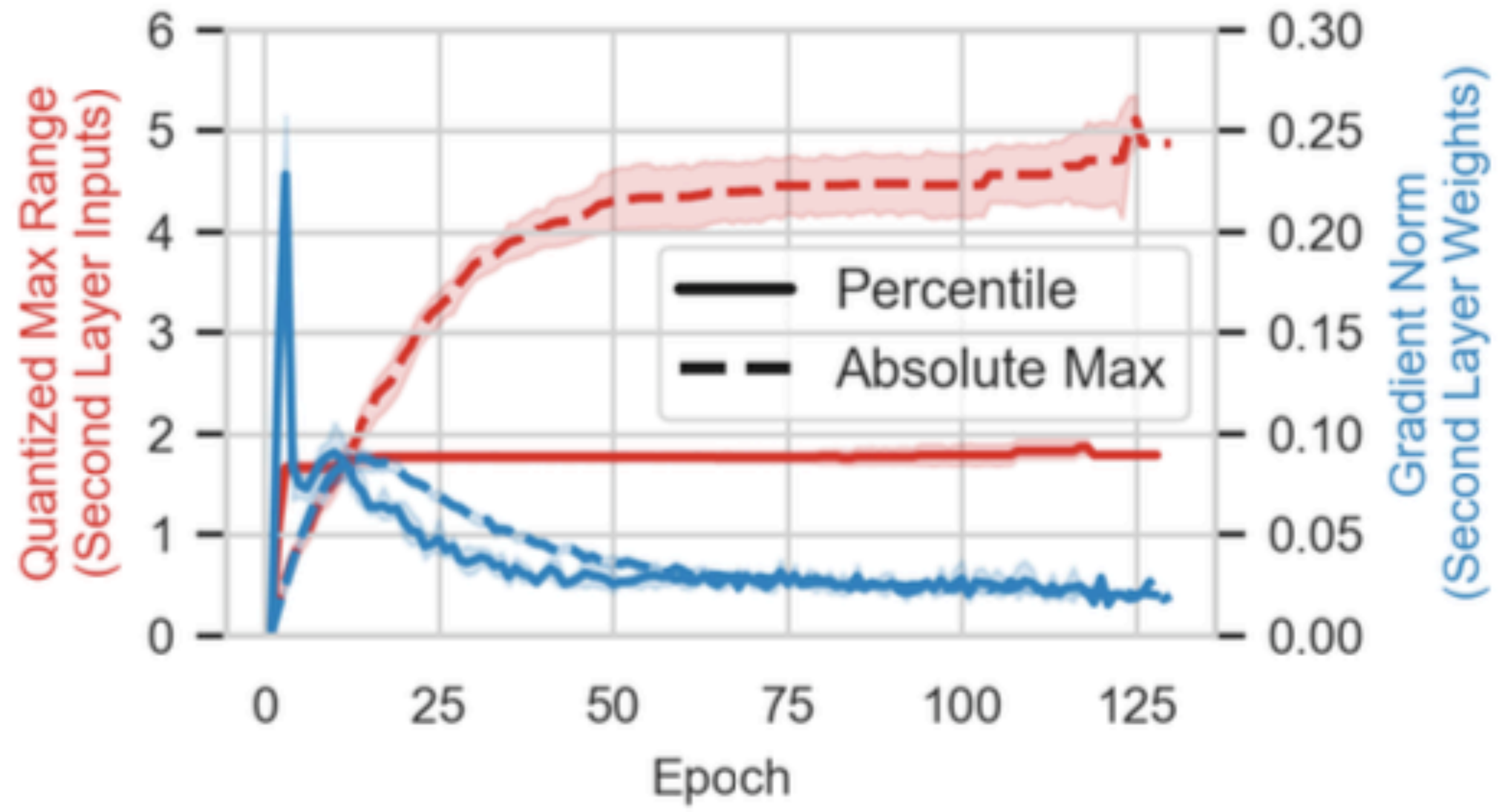

Figure 5: $q_{\max}$ with absolute min/max and percentile ranges, applied to INT8 GCN training on Cora. We observe that the percentile max is half that of the absolute, *doubling* resolution for the majority of values.

- Source of degradation in INT-4
- ***Aggregation and message***

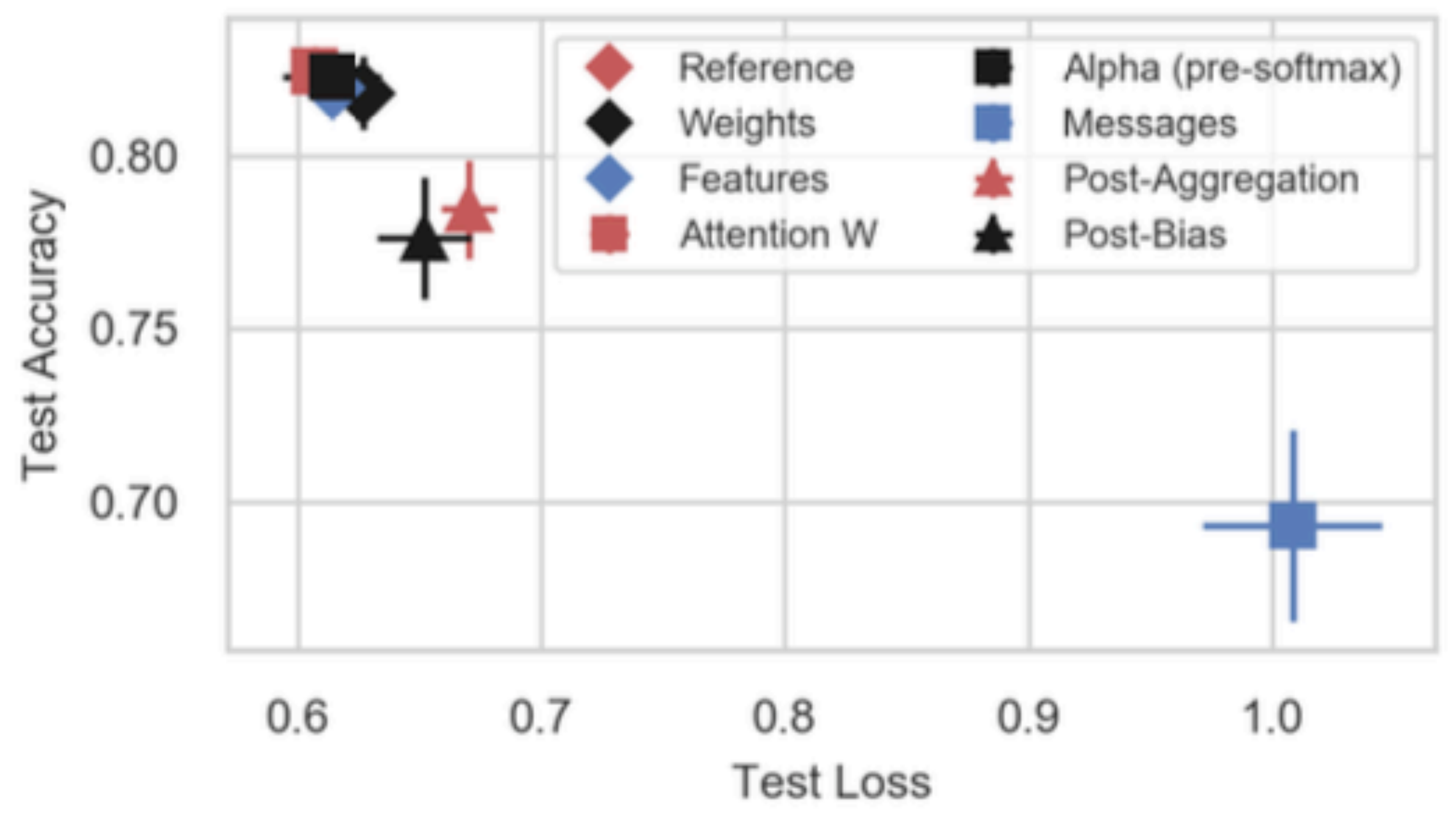

Figure 6: Analysis of how INT8 GAT performance degrades on Cora as individual elements are reduced to 4-bit precision *without DQ*. For GAT the message elements are crucial to classification performance.

Thanks