Quest: Stable Training of LLMs with 1-Bit Weight and Activation

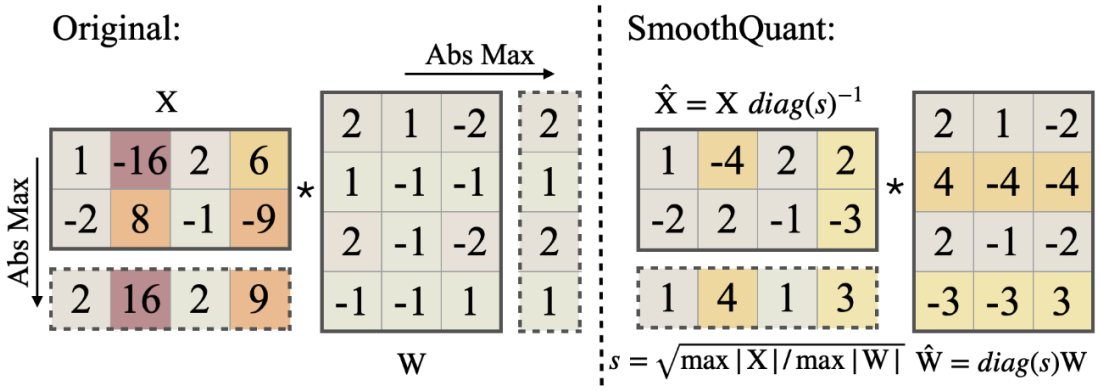
2025.05.07.

Computer Science and Engineering
POSTECH

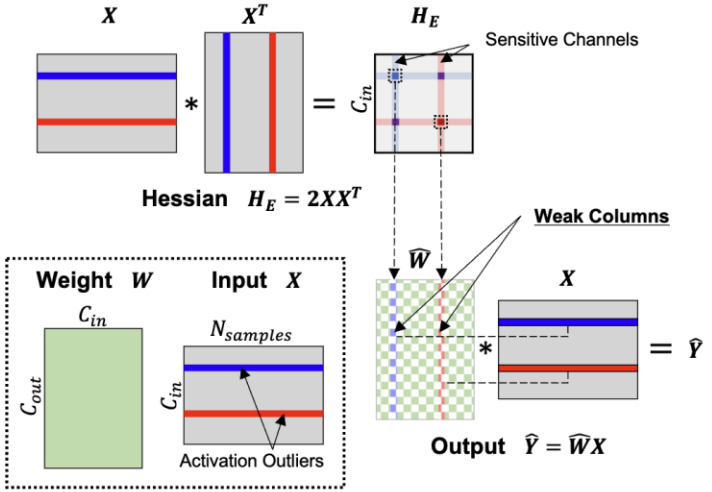
Introduction

- **PTQ is the most dominant method for LLM inference quantization.**
  - PTQ: Find the optimal quantization step size based on the statistical feature of pretrained weight.
  - QAT: Get an accurate discrete approximation of weight by optimizing parameters.

  - For LLM Inference Quantization, PTQ is the most popularly researched scheme.
    (SmoothQuant, GPTQ, AWQ, OWQ, etc…)



SmoothQuant



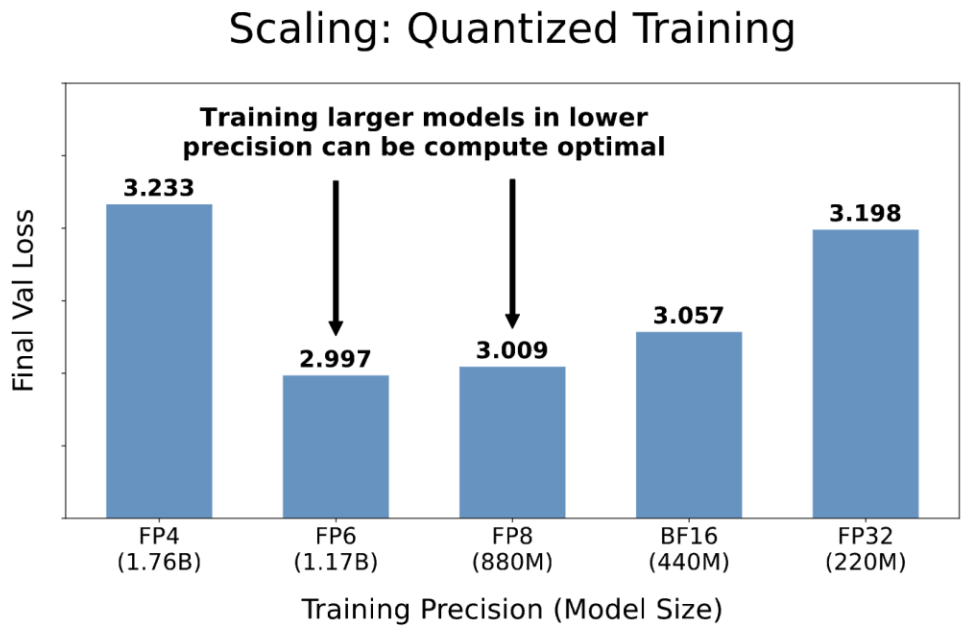
OWQ

Xiao et al., “SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models”

Lee et al., “OWQ: Outlier-Aware Weight Quantization for Efficient Fine-Tuning and Inference of Large Language Models”

Introduction

- **Why PTQ and not QAT so far?**
  - Generally, PTQ is faster and shows less computation than QAT.
  - So far, QAT shows high training costs and unstable convergence problems.
  - Kumar et al.(2024) insist that the current SOTA QAT shows the optimal points on 8-bit quantization.



Kumar et al., “Scaling Laws for Precision”

Introduction

- **Restriction of PTQ**
  - For hardware support for multiplication, both weight and activation extends down to 4-bit.
  - However, sota PTQ method still far from recovering full accuracy on W4A4

Table 3: WikiText-2 Perplexity and zero-shot accuracy of QuaRot on the LLAMA-2 family using 4- and 8-bits with Round-to-Nearest (RTN) weights and activation quantization. For zero-shot tasks, we use PIQA (PQ), WinoGrande (WG), HellaSwag (HS), Arc-Easy (A-e), Arc-Challenge (A-c), and LAMBADA (LA). We quantize all weights, activations, and caches.

Model	Method	Precision	PPL ↓	PQ ↑	WG ↑	HS ↑	A-e ↑	A-c ↑	LA ↑	Avg. ↑
7B	Baseline	FP16	5.47	79.11	69.06	75.99	74.58	46.25	73.90	69.82
	QuaRot-RTN	INT4	8.37	72.09	60.69	65.40	58.88	35.24	57.27	58.26
		INT8	5.50	78.94	68.67	75.80	74.79	45.39	74.33	69.65
70B	Baseline	FP16	3.32	82.70	77.98	83.84	80.98	57.34	79.58	77.07
	QuaRot-RTN	INT4	4.14	80.69	75.14	79.63	77.57	51.71	77.02	73.63
		INT8	3.33	82.97	77.98	83.67	80.77	58.11	79.53	77.17

QuaRot: W4A4, W8A8 results

Ashkboos et al., "QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs "

Introduction

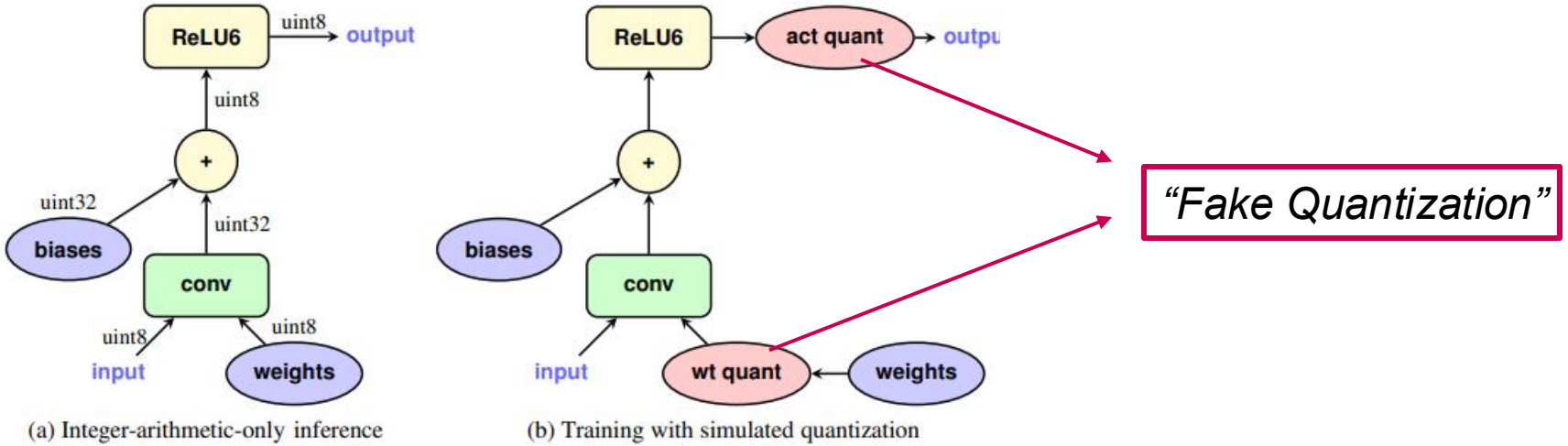
- **QAT has an apparent strength in accuracy.**
  - In the vision model, QAT outperformed the PTQ method due to its characteristics, "learn the optimal step size that effectively reflects the input's distribution".

The goal of this paper:

1. Overcome the Pareto-optimal bit-width(8-bit) of LLM-QAT method.
2. Achieve the super-low bit-width (1-bit) quantization without large errors in LLM.

Background

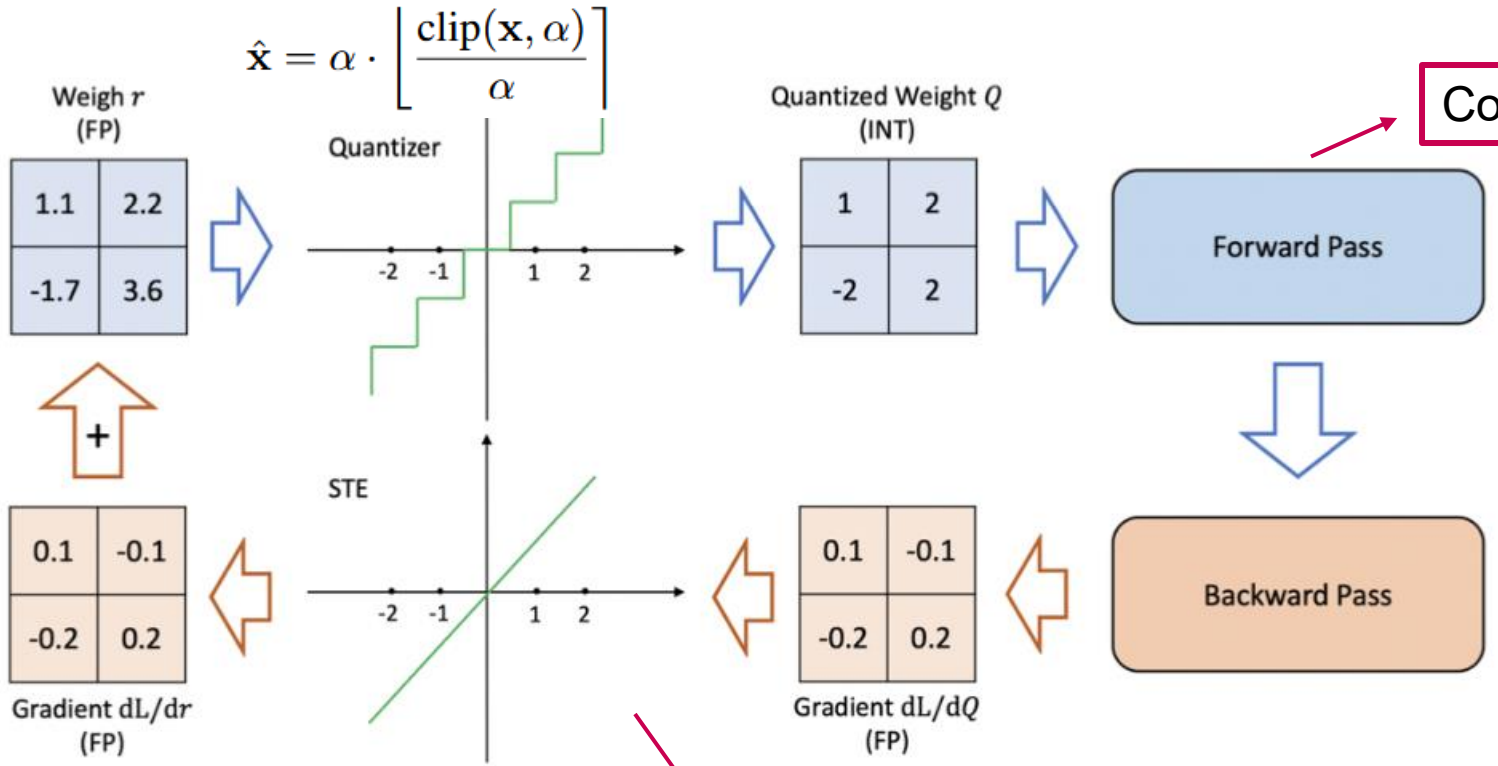
- QAT
  - Use 'fake quantization' operation to train quantization effects



Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference"

Background

- QAT
  - Use 'fake quantization' operation to train quantization effects
  - Clipping is not-differentiable, so use STE to get gradient.



Continuous-to-Discrete tensor fitting

Relying on STE, QAT learn the appropriate quantization

Background

- **Hadamard Matrix**
  - Hadamard Matrix: Orthogonal, all elements are +1 or -1
  - Inverse Hadamard matrix is transpose of itself

$$H_1 = [1],$$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

$$\mathrm{H}_n = \mathrm{H}_1 \otimes \mathrm{H}_{n-1},$$

$$M \otimes N = \begin{pmatrix} M_{11}N & \cdots & M_{1n}N \\ \vdots & & \vdots \\ M_{m1}N & \cdots & M_{mn}N \end{pmatrix}$$

Background

- **Hadamard Transformation**
  - Matmul with Hadamard Matrix, vector goes to Hadamard domain (linear transformation)

$$\tilde{v} = H_d \cdot v,$$

  - Few works report that "Hadamard Transformation leads **weight distribution match to Gaussian"**

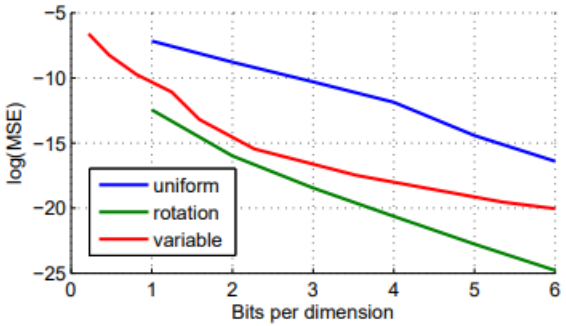
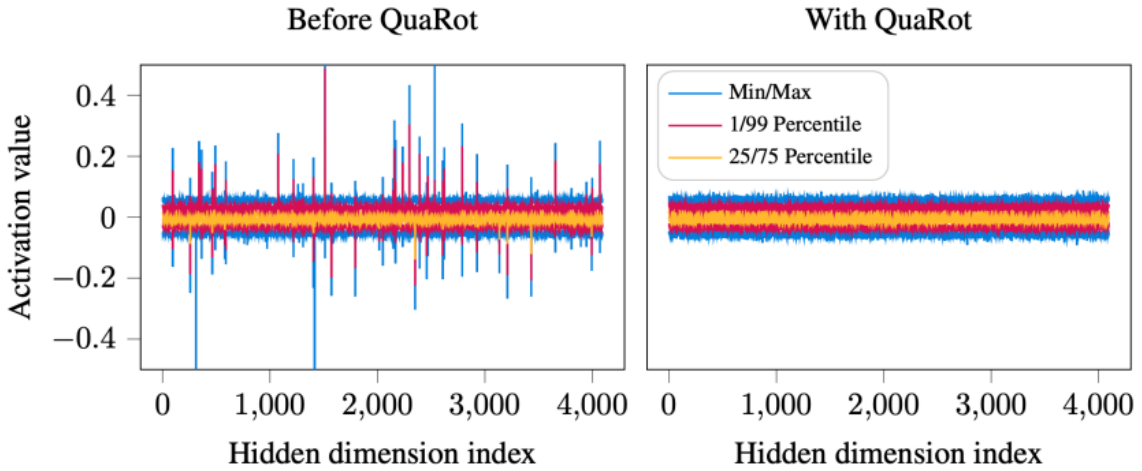


Figure 1: Distributed mean estimation on data generated from a Gaussian distribution.



Ashkboos et al., "QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs "

Suresh et al., "Distributed mean estimation with limited communication"

Background

- **Hadamard Transformation**
  - Matmul with Hadamard Matrix, vector goes to Hadamard domain (linear transformation)

$$\tilde{v} = H_d \cdot v,$$

  - Few works report that "Hadamard Transformation leads **weight distribution match to Gaussian"**

  - **Adbantages**
    - Inverse matrix is transpose of itself -> Easy to calcultate inverse matrix
    - Hadamard multiplication is supported by hardware
    - Hadamard domain distributes outlier, which is quantization-friendly

Method

- **Minimizing the quantization error of gradients.**
  - QAT: Approximate the discrete weight by optimizing parameters relying on STE.
  - However, it is difficult due to misalignment between discrete weight's gradient and continuous weight's gradient.
  - **Solution:** Selective gradient masking.
    - Big update for small quant error weight element.
    - Small update for large quant error weight element.
    - Update based on Discrete Mask of weight and activation.

$$\left\| \nabla_{\mathbf{w}} L - \nabla_{\hat{\mathbf{w}}} L \right\|_2^2. \tag{2}$$

Let us define the *quantization error* for each entry w_k as $\text{err}_k = \left| w_k - \hat{w}_k \right|$. We can partition the weight indices k based on whether the quantization error err_k is smaller or larger than some "trust factor" threshold T. Denote:

$$S_{\text{small}} = \{ k : \text{err}_k \leq T \}, \qquad S_{\text{large}} = \{ k : \text{err}_k > T \}.$$

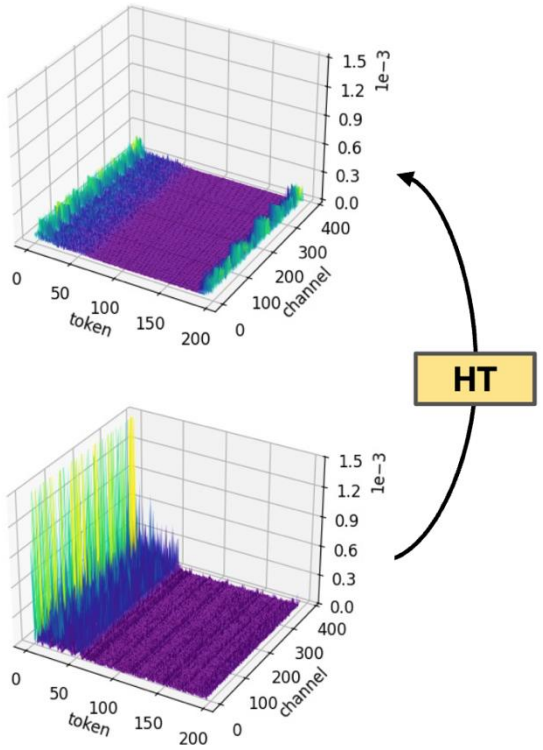
Then, the squared gradient difference in (2) decomposes as:

$$\underbrace{\sum_{k \in S_{\text{small}}} (\nabla_{\mathbf{w}} L_k - \nabla_{\hat{\mathbf{w}}} L_k)^2}_{(\star)} + \underbrace{\sum_{k \in S_{\text{large}}} (\nabla_{\mathbf{w}} L_k - \nabla_{\hat{\mathbf{w}}} L_k)^2}_{(\star\star)}.$$

Method

- **Forward: Hadamard Preprocessing**
  - Large quant error elements (S_{large}) should be eliminated as many as possible for stable convergence.
  - S_{large} is generally shown outlier values.
  - Suppress and mitigate the outlier value by Hadamard Transformation.

$$\hat{\mathbf{x}}_h = \mathrm{proj}_{\alpha^*} \boxed{\mathrm{HT}(\mathbf{x}).}$$



Kim et al., “HOT: Hadamard-based optimized training”

Method

- **Forward: Gaussian Fitting**
  - More strongly normalize the distribution of tensor via root mean square(RMS) normalization.
  - Align the distribution of tensor to $\mathcal{N}(0,1)$.
  - Find the quantization step size α^* to minimize the L2 error resulting from $\mathcal{N}(0,1)$.

$$\hat{\mathbf{x}}_h = \boxed{\text{proj}_{\alpha^*}}\text{HT}(\mathbf{x}).$$

$$\hat{\mathbf{x}} = \alpha^* \cdot \text{RMS}(\mathbf{x}) \cdot \left\lfloor \frac{\boxed{\text{clip}\left(\mathbf{x}/\text{RMS}(\mathbf{x}), \alpha^*\right)}}{\alpha^*} \right\rceil =$$

$$:= \text{proj}_{\alpha^*}(\mathbf{x}), \text{ where}$$

$$\alpha^* := \arg\min_{\alpha \in \mathbb{R}} \mathbb{E}_{\xi \sim \mathcal{N}(0,1)} \boxed{\left\| \xi - \alpha \cdot \left\lfloor \frac{\text{clip}(\xi, \alpha)}{\alpha} \right\rceil \right\|_2^2}$$

Method

- **Backward: Trust Gradient Estimation**

  - Element-wise product the mask resulting from the tensor's Gaussian Fitting with gradient.
  - Conduct Inverse Hadamard Transformation (IHT)
    - Return to original space from Hadamard space.
    - Discrete mask is converted to a continuous domain via IHT.
    - S_{large} elements can participate the subtle gradient update, so can make the convergence more stable.

$$\frac{\partial}{\partial \mathbf{x}} \approx \boxed{\mathrm{IHT}}\left(\boxed{M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h) \odot \frac{\partial}{\partial \hat{\mathbf{x}}_h}}\right).$$

Method

- Algorithm of QuEST

Hadamard Preprocessing

Gaussian Fitting

Algorithm 1 QuEST Training Forward

Input: Input activations $\mathbf{x}$, row-major weight $\mathbf{w}$
$\mathbf{x}_h = \mathrm{HT}(\mathbf{x})$
$\hat{\mathbf{x}}_h = \mathrm{proj}_{\alpha^*}\, \mathbf{x}_h$
$\mathbf{w}_h = \mathrm{HT}(\mathbf{w})$
$\hat{\mathbf{w}}_h = \mathrm{proj}_{\alpha^*}\, \mathbf{w}_h$
$\mathbf{y} = \hat{\mathbf{x}}_h \hat{\mathbf{w}}_h^T$
Return: $\mathbf{y}, \hat{\mathbf{x}}_h, \hat{\mathbf{w}}_h, M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h), M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h)$

Trust Gradient Estimation

Algorithm 2 QuEST Training Backward

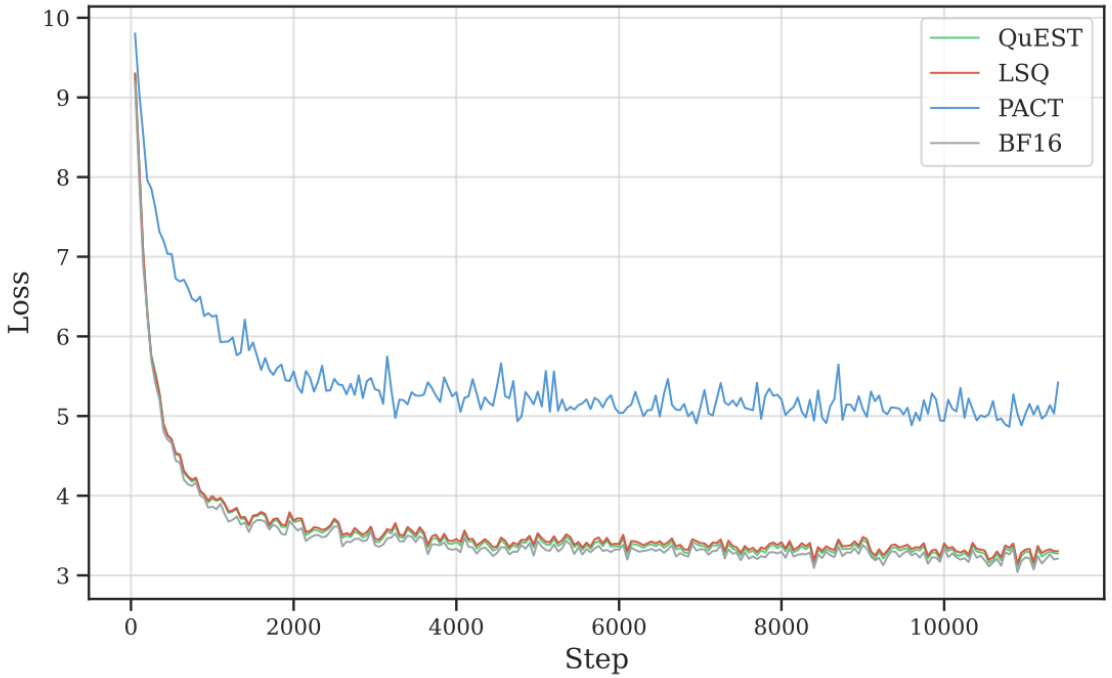
Input: $\frac{\partial L}{\partial \mathbf{y}}, \hat{\mathbf{x}}_h, \hat{\mathbf{w}}_h, M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h), M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h)$
$\frac{\partial L}{\partial \hat{\mathbf{x}}_h} = \frac{\partial L}{\partial \mathbf{y}} \hat{\mathbf{w}}_h$
$\frac{\partial L}{\partial \mathbf{x}} = \mathrm{IHT}\left(M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h) \odot \frac{\partial L}{\partial \hat{\mathbf{x}}_h}\right)$
$\frac{\partial L}{\partial \hat{\mathbf{w}}_h} = \hat{\mathbf{x}}_h^T \frac{\partial L}{\partial \mathbf{y}}$
$\frac{\partial L}{\partial \mathbf{w}} = \mathrm{IHT}\left(M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h) \odot \frac{\partial L}{\partial \hat{\mathbf{w}}_h}\right)$
Return: $\frac{\partial L}{\partial \mathbf{x}}, \frac{\partial L}{\partial \mathbf{w}}$

Experiment

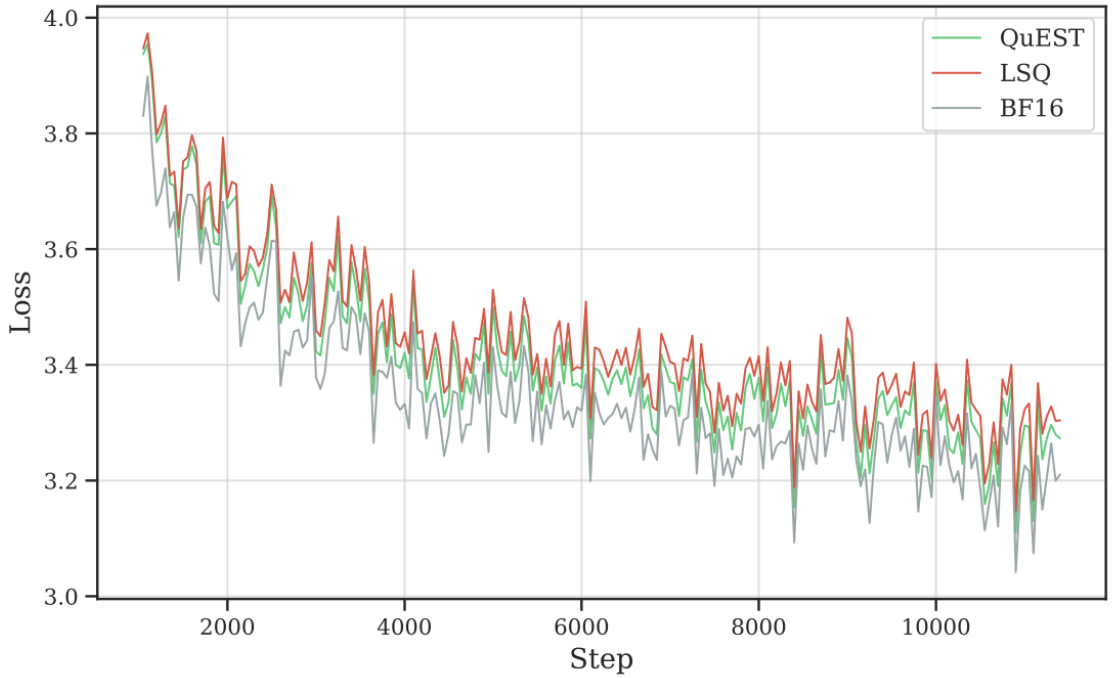
- **LLM experiment Setup**
  - Llama-family model (30M ~ 800M)
  - Pretraining C4 dataset
  - each model trained by (Free param * 100) token, regardless of precision
    - Ex : 100M model : 10B tokens
  - Optimizer : AdamW
  - Lr scheduler : Cosine decay, 10% warm up
  - Gradient clipping : threshold 1.0
  - Weight decay : decoupled, 0.1

Experiment

- **Comparison to Prior QAT Method.**
  - **Target : PACT, LSQ**
  - 30M model, 3B token W4A4



(a)



(b)

Experiment

- Comparison to Prior QAT Method.
  - Target : PACT, LSQ

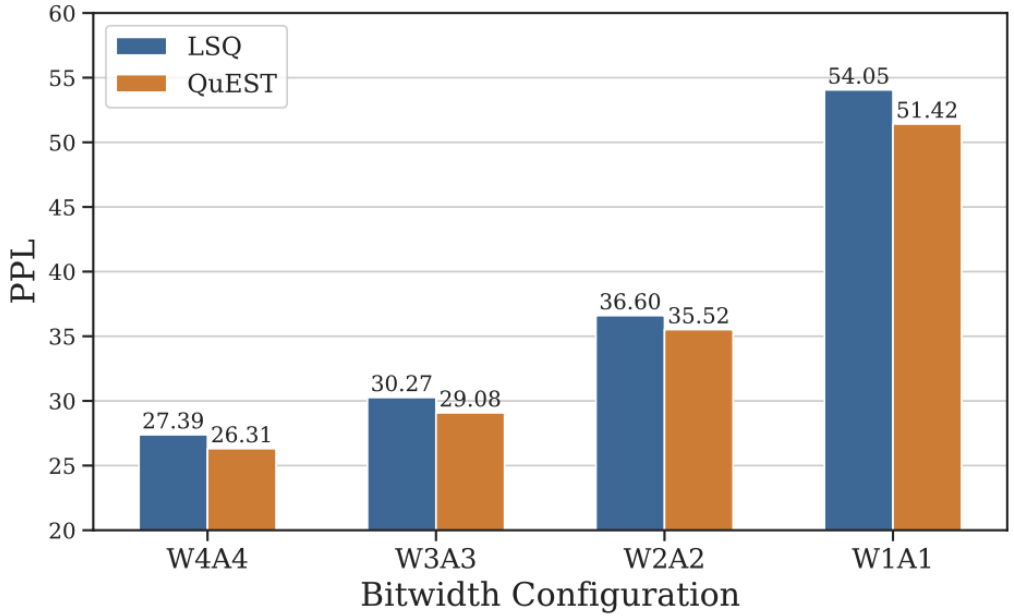


Figure 3. Perplexity (PPL) across bit-widths with QuEST vs. a tuned variant of LSQ on a 30M model. QuEST leads to consistently lower PPL, with the advantage growing with compression.

Experiment

- **LLM experiment result**
  - Llama-family model (30M, ~ 800M)
  - Pretraining C4 dataset
  - Demonstrate that:
    - Stable training in W1A1
    - W4A4 has Pareto-dominant relative to BF16

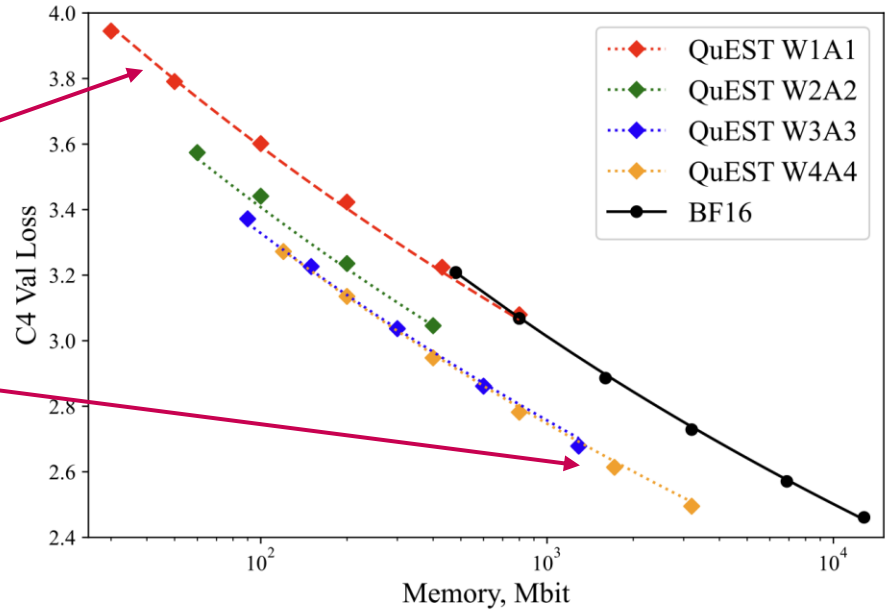


Figure 1. The scaling law induced by QuEST when training Llama-family models from 30 to 800M parameters on C4, with quantized weights and activations from 1 to 4 bits, in the 100 tokens/parameter regime (higher compression uses proportionally more data at fixed memory). QuEST allows for stable training at 1-bit weights and activations (W1A1), and the QuEST W4A4 model is Pareto-dominant relative to BF16, with lower loss at lower size.

Experiment

- **Scaling laws:**
  - Hoffman et al.(2022)
    - Loss can be modeling using the following parametric form:

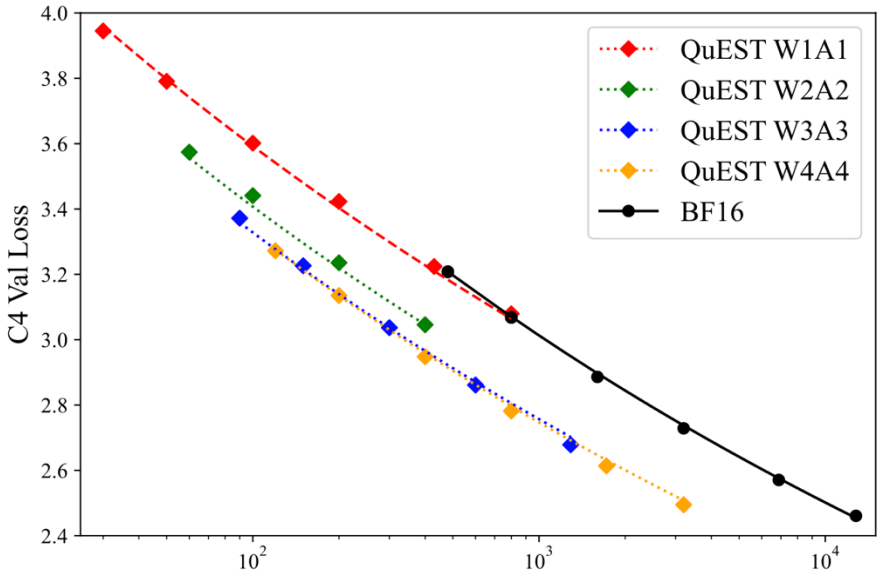
$$L(N, D) = AN^{-\alpha} + BD^{-\beta} + E,$$

    - N: number of model param
    - D: number of training tokens
    - A, B, E, α, β : Empirical coefficients fitted from data

Experiment

- **Scaling laws**
  - To model quantized training, the original formula is extended to account for precision P:

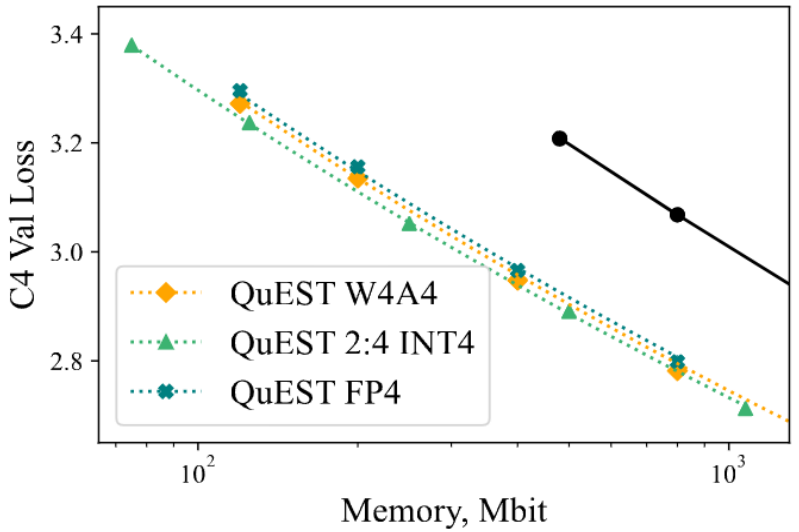
$$L(N, D, P) = \frac{A}{(N \cdot \text{eff}(P))^{\alpha}} + \frac{B}{D^{\beta}} + E.$$



  - eff(P): **Effective parameter multiplier** for precision PP
  - At P=16, eff(P)=1.0 recovers the original FP16 formulation
  - the scaling law parameters (A, B, E, α, β, eff(P)) are fitted via **Huber loss** regression over observed loss values across different (N,D,P)(N,D,P) combinations.

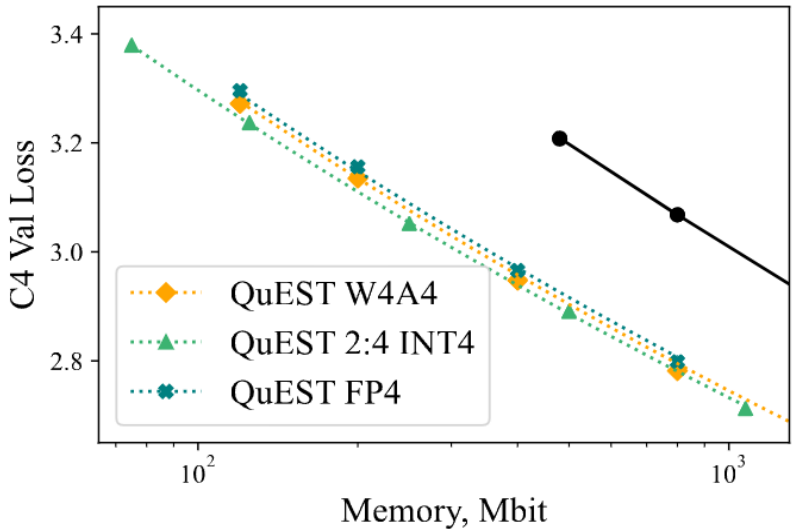
Experiment

- **Extensions to Different Formats (FP4)**
  - QuEST is extended to FP4 by replacing the uniform rounding operation with rounding to the FP4 grid scaled to $[-1, 1]$
  - The trust mask is adjusted based on the widest interval in the FP4 grid.
  - FP4 performs slightly worse than INT4 in terms of parameter efficiency.
  - This may be due to higher MSE when fitting Gaussian data under clipping



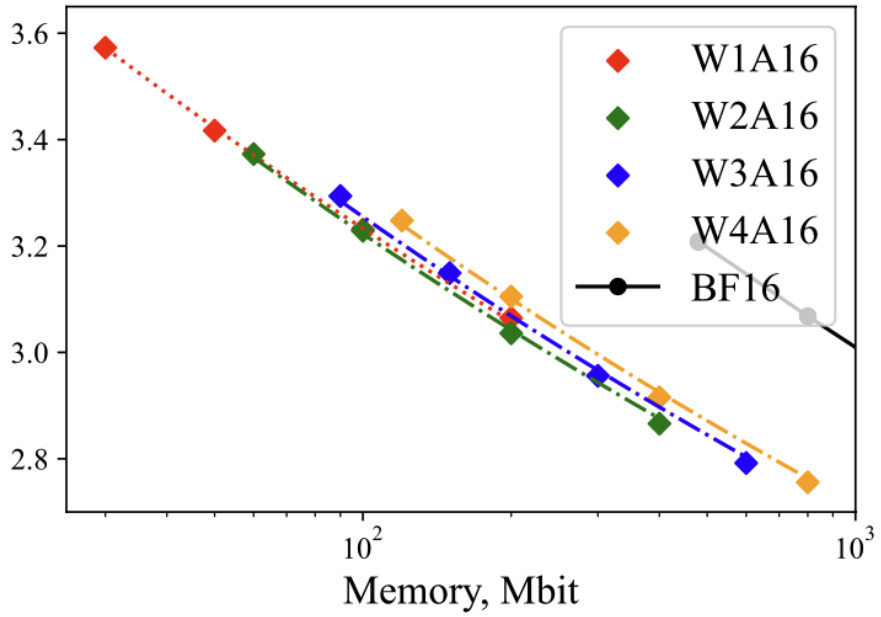
Experiment

- **Extensions to Different Formats (2:4 Sparsity)**
  - QuEST also supports sparse weights.
    - During the forward pass: sparsify → quantize.
    - During the backward pass: apply the trust mask as usual
  - outperforms FP4 but is slightly less efficient than INT4.



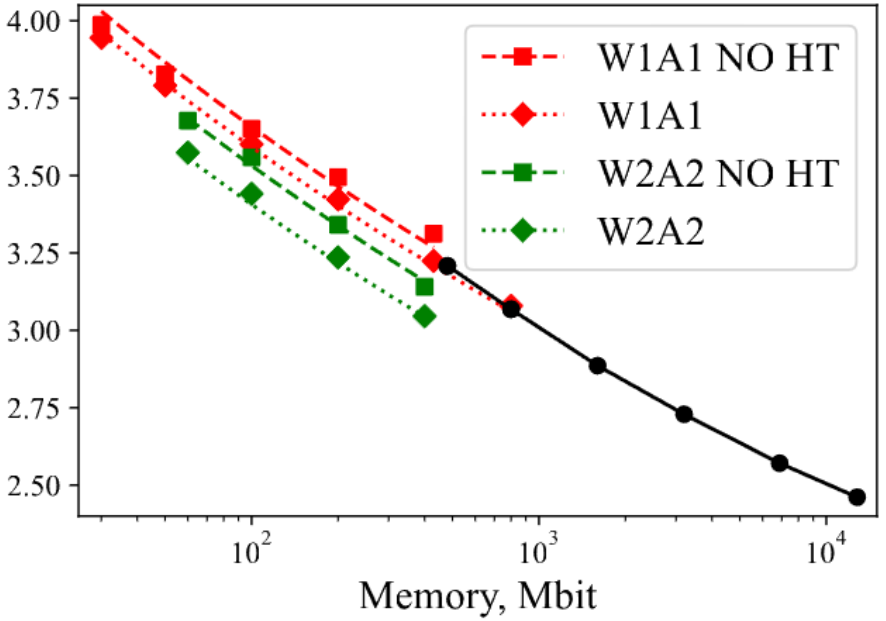
Experiment

- **Other variation: weight-only-quantization**
  - train 30–200M parameter models using only weight quantization at 1–4 bits.
  - 2-bit weights are Pareto-optimal.
  - 1-bit weights surprisingly outperform 3-bit in some cases.



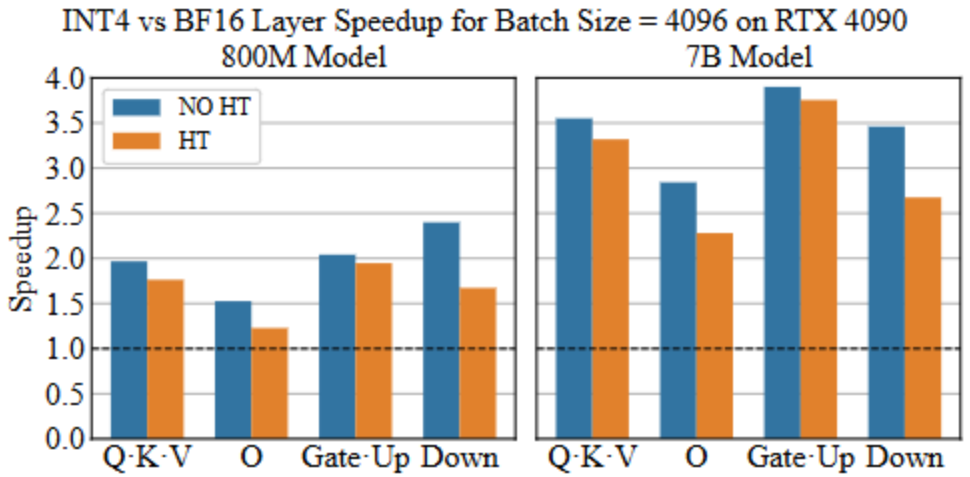
Experiment

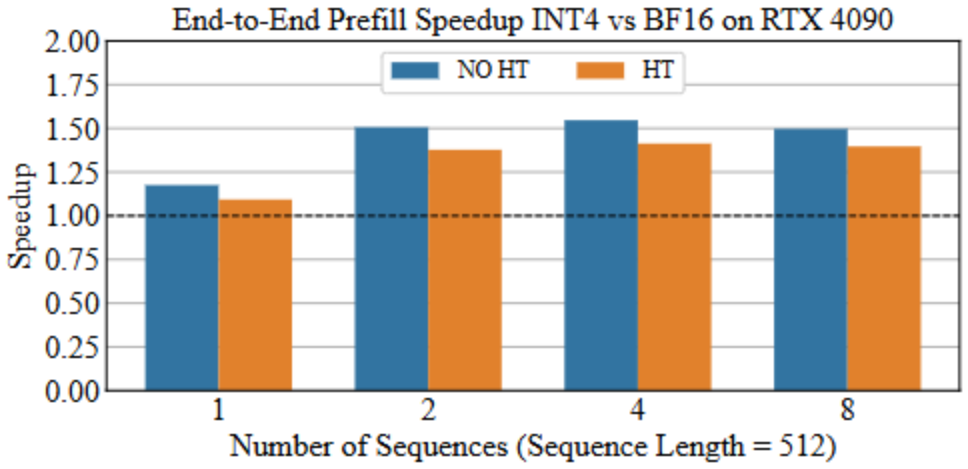
- **Hadamard Transform Ablation**
  - Training remains stable across all bitwidths.
  - W1A1 (1-bit weights & activations) underperforms compared to BF16.
  - W4A4 remains Pareto-optimal, suggesting the Hadamard step improves efficiency but does not change scaling laws.



Experiment

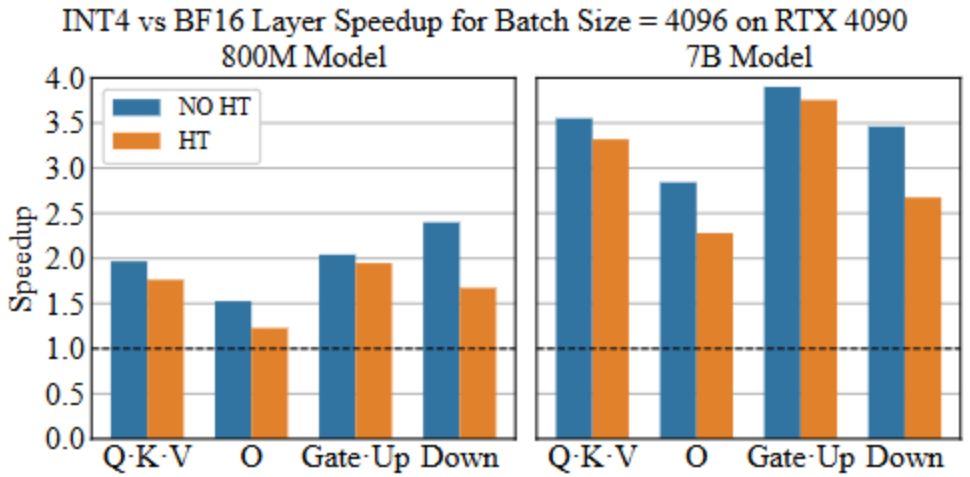
- **GPU Execution for QuEST Models (Layer wise, left)**
  - For the 800M model:
    - Up to 1.2× speedup on the smallest layers (with Hadamard)
    - Up to 2.4× speedup on the largest down-projection layer (without Hadamard)
    - Hadamard overhead reaches up to 30% on the down-projection layer
  - For the 7B model: 2.3× to 3.9× speedups due to larger compute per layer.

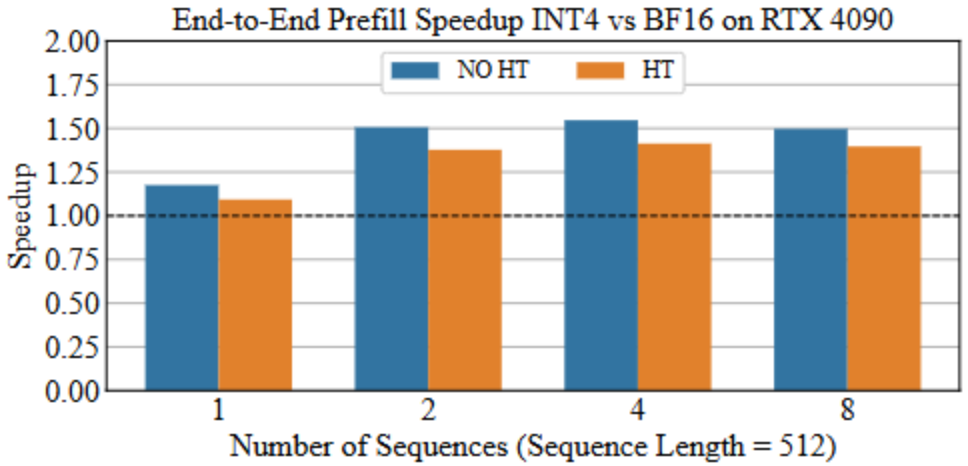




Experiment

- **GPU Execution for QuEST Models (end to end , right)**
  - QuEST kernels demonstrate a 1.3× to 1.5× speedup over the BF16 baseline.





Discussion & Limitation

- **Unclear effect of scaling after Hadamard Transform**
  - HT already mitigates outliers; further Gaussian scaling may yield limited additional benefit
- **Gradient masking strategy lacks broader comparison**
  - QuEST favors large updates to low-error weights
  - Competing methods (e.g., INT4 Transformer) use random masking for unbiased updates
- **No comparison with Post-Training Quantization (PTQ)**
  - Cost–performance trade-offs between QAT vs. PTQ not addressed
  - Performance gains over PTQ methods remain unquantified
- **Limited to small-scale models**
  - Largest model tested is 800M; scalability of QAT to LLMs (>7B) remains uncertain

Thank you.