PE-QAT: Parameter-Efficient Quantization-Aware Training for Large Language Models

Shresth Mishra
Lynbrook High School
San Jose, CA
shresth.mishra@gmail.com

Abstract

As large language models (LLMs) grow, their compute and memory demands become prohibitive for on-device deployment. Quantization is a crucial technique to shrink model footprint and accelerate inference, but aggressively low-bit weight-activation quantization schemes often sacrifice accuracy. Quantization Aware Training (QAT) is a commonly used paradigm to minimize quantization noise, but is extremely expensive to train and often unscalable to large models. We introduce PE-QAT, a parameter-efficient framework targeting per-channel 4-bit weight-activation quantization of LLMs, which aims to preserve model accuracy while significantly reducing resource requirements. The proposed method freezes the base model and trains lightweight LoRA adapters by fake quantizing the merged-weight model, enabling PE-QAT to scale efficiently unlike full QAT. We apply fake quantization with Straight-Through Estimators (STE) to the merged weights, allowing the adapters to explicitly compensate for quantization noise during training. One of the biggest challenges with quantizing activations alongside weights is addressing outliers that are orders of magnitude larger than other activations, which inflate quantization scales and suppress lower-magnitude values. To mitigate the impact of severe activation outliers, PE-QAT jointly learns per-channel smoothing factors and symmetric activation clipping thresholds. PE-QAT retains accuracy within 0.11 percentage points of the full-precision baseline on Llama-2-7B zero-shot tasks while training only 1.26% of total parameters.

1 Introduction

Resource constraints are a recurring challenge in real-world deep learning deployment, including edge-deployed vision systems (Mishra et al., 2025) and large language models (Touvron et al., 2023; Brown et al., 2020). For LLMs in particular, deployment is increasingly bottlenecked by memory bandwidth and capacity, making 4-bit quantization a standard efficiency target. However, naive low-bit quantization, particularly of activations, often leads to severe performance degradation (Dettmers et al., 2022; Frantar et al., 2023) due to outliers that are orders of magnitude larger than the rest of the activations, driving up scales and effectively suppressing the majority of the activation values. Current solutions typically fall into two categories: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ methods like GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) utilize efficient calibrations but can struggle to recover accuracy in aggressive W4A4 settings. Full QAT (Jacob et al., 2018; Esser et al., 2020), while capable of high accuracy, is computationally expensive, often exceeding that of full fine tuning. In this work, we introduce PE-QAT, a resource-efficient post-training framework that targets per-channel W4A4 quantization. Instead of updating the massive base model, PE-QAT freezes the full-precision base weights, injects trainable Low-Rank Adapters (LoRA) (Hu et al., 2022) to compensate for quantization error, quantizes the merged model (avoiding the overhead from mixed precision inference), and learns per-channel SmoothQuant-style (Xiao et al., 2023) smoothing factors and symmetric clipping thresholds during backpropagation to address activation outliers. PE-QAT reduces training costs compared to full QAT by training only 1% of the total trainable parameters, while providing the quantized model with more flexibility to adapt to quantization noise than PTQ methods. We demonstrate that PE-QAT loses only 0.11 accuracy compared to the full precision accuracy on zero-shot commonsense reasoning tasks for Llama-2-7B, offering a practical middle ground between the accessibility of PTQ and the performance of full QAT.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 701–711

2 Background

2.1 Quantization

Uniform symmetric quantization (Jacob et al., 2018; Gholami et al., 2021) maps floating-point weights W to a discrete set of integer levels. For b-bit quantization, we define the quantization range as $Q_{\min} = -2^{b-1}$, $Q_{\max} = 2^{b-1} - 1$.

The scale factor (step size between quantization levels) is computed as:

$$s = \frac{\max(|W|)}{Q_{\max}}$$

Quantization is performed by dividing by the scale, rounding, and clipping:

$$w_q = \text{clip}\left(\left\lfloor \frac{W}{s} \right\rceil, Q_{\min}, Q_{\max}\right)$$

where $w_q \in \mathbb{Z}$ are the quantized integer weights. Dequantization recovers the floating-point approximation by multiplying by the scale:

$$\tilde{W} = w_q \cdot s$$

Weight-only quantization replaces the standard forward pass $Y = WX$ with $\tilde{Y} = \tilde{W}X$, still requiring floating point matrix multiplication. To enable full integer arithmetic, weight-activation quantization is performed using the same procedure with scale $s_x = \max(|X|)/Q_{\max}$:

$$\tilde{Y} = s_w \cdot s_x \cdot (w_q \cdot x_q)$$

where the integer matrix multiplication $w_q \cdot x_q$ can be performed efficiently, and the scales are applied as a single floating-point multiplication at the end.

2.2 LoRA

Rather than updating entire model weights during training, Low-Rank Adapters (LoRA) (Hu et al., 2022) inject small trainable rank decomposition matrices while freezing the original weights. Given a weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA introduces two low-rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, where $r \ll \min(d, k)$.

$$W = W_0 + \frac{\alpha_{\text{LoRA}}}{r} BA$$

Where α_{LoRA} is a hyperparameter that controls the influence of the adaptors.

$$W = W_0 + \Delta W$$

where $\Delta W \in \mathbb{R}^{d\times k}$. Low Rank Adapters thus reduce the number of trainable parameters from $d \times k$ to $r \times (d+k)$, enabling efficient adaptation of large models with minimal resource requirements, as r is kept at a much smaller value than d and k.

3 Proposed Method

We introduce Parameter-Efficient Quantization-Aware Training (PE-QAT), a framework that delivers a middle ground between the efficiency of PTQ and performance of QAT. PE-QAT avoids the prohibitive memory cost of standard QAT by freezing the full-precision base model and injecting trainable parameters only through LoRA and lightweight quantization parameters. We quantize weights and activations to 4 bits using per-channel symmetric quantization, and keep the KV-cache in full precision. We apply PE-QAT to the attention projections and MLP projections in each transformer block.

3.1 Architecture Overview

Algorithm 1 presents the complete PE-QAT forward pass for a single linear layer. Given pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ (where d is the output dimension and k is the input dimension) and input activations $X \in \mathbb{R}^{N\times k}$ (where N is batch size $\times$ sequence length), PE-QAT merges LoRA adapters, applies learned smoothing, clips activations, and performs fake quantization with Straight-Through Estimators (STE) for gradient flow. In our notation, weight matrix W is organized with output channels as rows and input channels as columns, so the linear transformation is computed as $Y = XW^T$ where $W^T \in \mathbb{R}^{k\times d}$ is the transposed weight matrix. Current implementation uses fake quantization; real INT4 inference requires additional deployment optimization. At inference, the merged weights W_{eff} would be quantized to INT4 and scales pre-computed, enabling efficient integer matrix multiplication $(X_q)(W_q)^T$ with post-hoc scaling.

3.2 Learnable Activation Outlier Smoothing

Activation outliers in LLMs (Dettmers et al., 2022) (e.g., Llama-2 (Touvron et al., 2023)) make static W4A4 quantization difficult. Instead of using fixed calibration scales, PE-QAT uses learnable per-input-channel smoothing factors $s_{\text{smooth}} \in \mathbb{R}^k$.

The smoothing operation applies per-input-channel scaling to migrate quantization difficulty

Algorithm 1 PE-QAT Forward Pass (per linear layer)

Require: Pre-trained weight $W_0 \in \mathbb{R}^{d\times k}$, input $X \in \mathbb{R}^{N\times k}$
Require: LoRA adapters $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, scaling α_{LoRA}, rank r
Require: Learned smoothing $s_{\text{smooth}} \in \mathbb{R}^k$, clipping $\alpha_{\text{clip}} \in \mathbb{R}^k$
Require: Quantization bits $b = 4$, $Q_{\min} = -2^{b-1}$, $Q_{\max} = 2^{b-1} - 1$
Ensure: Quantized output $Y \in \mathbb{R}^{N\times d}$

// Step 1: Merge LoRA with base weights
$W_{\text{eff}} \leftarrow W_0 + \frac{\alpha_{\text{LoRA}}}{r} BA$
// Step 2: Apply per-channel smoothing (vectorized over columns)
$W'^{(:,c)} \leftarrow W_{\text{eff}}^{(:,c)} \cdot s_{\text{smooth}}^{(c)}$ for $c = 1, \ldots, k$
$X'^{(:,c)} \leftarrow X^{(:,c)} / s_{\text{smooth}}^{(c)}$ for $c = 1, \ldots, k$
// Step 3: Clip activations per input channel (vectorized over rows)
$X_{\text{clipped}}^{(:,c)} \leftarrow \text{clip}(X'^{(:,c)}, -\alpha_{\text{clip}}^{(c)}, \alpha_{\text{clip}}^{(c)})$ for $c = 1, \ldots, k$
// Step 4: Quantize weights per output channel (vectorized over columns)
for $j = 1$ to d **do**
  $s_w^{(j)} \leftarrow \max_{c\in[1,k]} |W'^{(j,c)}| / Q_{\max}$
  $W_q^{(j,:)} \leftarrow \text{clip}\left(\text{round}(W'^{(j,:)} / s_w^{(j)}), Q_{\min}, Q_{\max}\right)$
  $\tilde{W}^{(j,:)} \leftarrow W_q^{(j,:)} \cdot s_w^{(j)}$
end for
// Step 5: Quantize activations per input channel (vectorized over rows)
for $c = 1$ to k **do**
  $s_x^{(c)} \leftarrow \alpha_{\text{clip}}^{(c)} / Q_{\max}$
  $X_q^{(:,c)} \leftarrow \text{clip}\left(\text{round}(X_{\text{clipped}}^{(:,c)} / s_x^{(c)}), Q_{\min}, Q_{\max}\right)$
  $\tilde{X}^{(:,c)} \leftarrow X_q^{(:,c)} \cdot s_x^{(c)}$
end for
// Step 6: Compute output via transposed matrix multiplication
$Y \leftarrow \tilde{X}\tilde{W}^T$ ▷ $Y \in \mathbb{R}^{N\times d}$, gradients flow via STE
Return Y

from activations to weights. Specifically, for each input channel $c \in [1, k]$, we scale the corresponding column in both the weight matrix and activation matrix:

$$W'^{(j,c)} = W_{\text{eff}}^{(j,c)} \cdot s_{\text{smooth}}^{(c)}, \quad \forall j \in [1, d] \tag{1}$$

$$X'^{(t,c)} = X^{(t,c)} / s_{\text{smooth}}^{(c)}, \quad \forall t \in [1, N] \tag{2}$$

This operation preserves the linear transformation since:

$$\begin{aligned} Y &= XW^T \\ &= X' \,\text{diag}(s_{\text{smooth}})\ \text{diag}(s_{\text{smooth}})^{-1} (W')^T \\ &= X'(W')^T. \end{aligned} \tag{3}$$

where $\text{diag}(s_{\text{smooth}}) \in \mathbb{R}^{k\times k}$ is a diagonal matrix with $s_{\text{smooth}}^{(c)}$ on the diagonal. By updating s_{smooth} during training via gradient descent, the model dynamically learns the optimal per-channel scale to suppress activation outliers, making X' easier to quantize while shifting the quantization difficulty to weights W', which typically have a more favorable distribution for quantization. We initialize the smoothing factors using a percentile-based variant of SmoothQuant (Xiao et al., 2023). Let $\mathcal{X}_c = \{X^{(s,t,c)} : s \in [1, B_{\text{cal}}], t \in [1, T]\}$ denote the set of activation values for input channel c collected over B_{cal} calibration batches with sequence length T. Let $\mathcal{W}_c = \{W_0^{(j,c)} : j \in [1, d]\}$ denote the set of weight values in column c of the pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$.

We define per-channel statistics:

$$A_c = \text{quantile}_{0.995}(|\mathcal{X}_c|), \qquad B_c = \max(|\mathcal{W}_c|) \tag{4}$$

and initialize the per-channel smoothing factor as:

$$s_{\text{smooth}}^{(c)} = \frac{A_c^\lambda}{(B_c + \epsilon)^{1-\lambda}}, \quad c = 1, \ldots, k \tag{5}$$

with $\lambda = 0.5$ and $\epsilon = 10^{-6}$ for numerical stability. The 99.5th percentile is more robust to extreme outliers than the maximum, especially with limited calibration data. This initialization balances the dynamic ranges of activations and weights: channels with large activation outliers receive larger smoothing factors, which suppress the activations while amplifying the corresponding weight column.

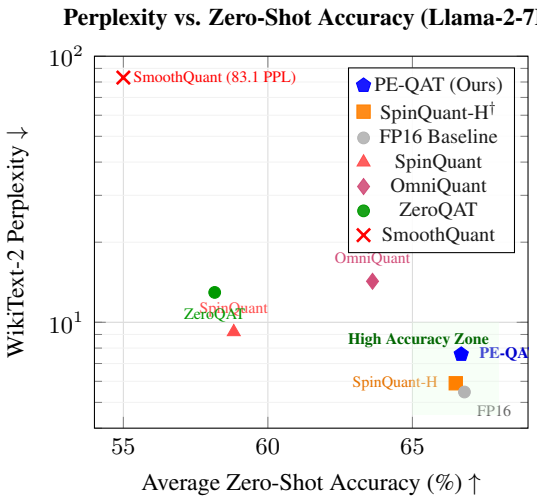

Figure 1: Perplexity vs. Accuracy Tradeoff. PE-QAT (blue) achieves slightly higher accuracy (+0.19%) than SpinQuant-H (orange) without requiring Hadamard transformations, despite the perplexity difference. Both methods sit in the "High Accuracy Zone" alongside the FP16 baseline, far separated from prior PTQ methods.

3.3 Learnable Activation Clipping

PE-QAT incorporates learnable activation clipping to address the quantization-clipping trade-off. We

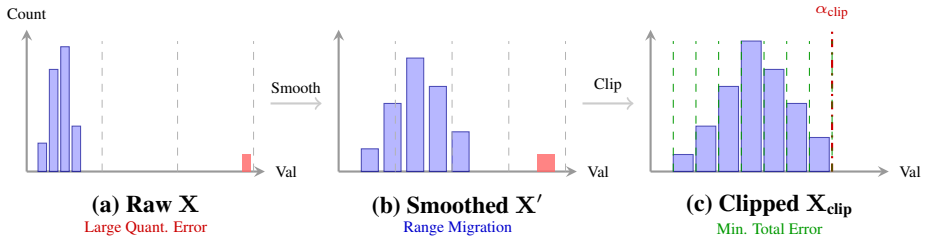

Figure 2: The PE-QAT activation processing pipeline. (a) Raw activations contain extreme outliers forcing a high quantization scale. (b) Learnable smoothing migrates outliers into the weight distribution. (c) Learnable clipping eliminates remaining outliers, allowing for high-precision quantization bins within the clipping boundary α_{clip}.

use per-input-channel learnable clipping thresholds $\alpha_{\text{clip}} \in \mathbb{R}^k$, initialized at the 99.5th percentile of the activation distribution observed during calibration:

$$\alpha_{\text{clip}}^{(c)} = \text{quantile}_{0.995}(|\mathcal{X}_c|), \quad c = 1, \ldots, k \quad (6)$$

where $\mathcal{X}_c = \{X^{(s,t,c)} : s \in [1, B_{\text{cal}}], t \in [1, T]\}$ denotes the activations for input channel c collected over calibration batches and tokens (as defined in Section 3.2). This percentile-based initialization ensures that only 0.5% of extreme outliers are clipped initially, preserving the majority of the activation range. During training, $\alpha_{\text{clip}}^{(c)}$ is constrained to be positive via $\max(\alpha_{\text{clip}}^{(c)}, \epsilon)$ where $\epsilon = 10^{-6}$ for numerical stability.

During forward passes, smoothed activations X' are symmetrically clipped per-channel. For each token $t \in [1, N]$ and channel $c \in [1, k]$:

$$X_{\text{clipped}}^{(t,c)} = \text{clip}(X'^{(t,c)}, -\alpha_{\text{clip}}^{(c)}, \alpha_{\text{clip}}^{(c)}) \quad (7)$$

The clipped activations are then quantized using a per-channel symmetric quantization scale:

$$s_x^{(c)} = \frac{\alpha_{\text{clip}}^{(c)}}{Q_{\text{max}}}, \quad c = 1, \ldots, k \quad (8)$$

where $Q_{\text{max}} = 2^{b-1} - 1 = 7$ for 4-bit quantization. The clipping operation in (7) is non-differentiable, requiring custom gradients for $\alpha_{\text{clip}}^{(c)}$. We decompose $\partial L / \partial \alpha_{\text{clip}}^{(c)}$ into two components inspired by PACT (Choi et al., 2018) and LSQ (Esser et al., 2020). The boundary gradient expands $\alpha_{\text{clip}}^{(c)}$ when clipped activations have large loss gradients, reducing information loss from hard clipping. Quantization scale gradient adjusts the quantization bin size $s_x^{(c)}$ to minimize the reconstruction error $\epsilon^{(t,c)} = \tilde{x}^{(t,c)} - x_{\text{clipped}}^{(t,c)}$, where $\tilde{x}^{(t,c)}$ is the dequantized activation. Complete derivations with explicit formulas are provided in Appendix A.3. To ensure

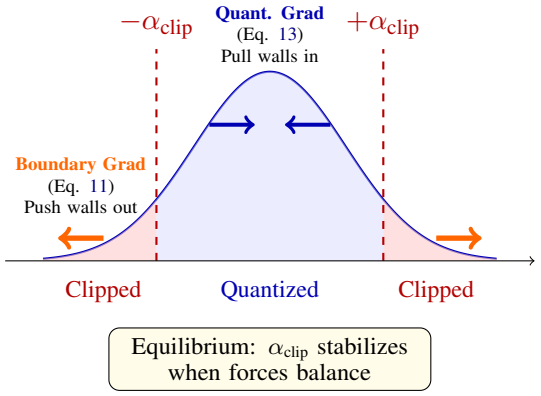

Figure 3: Dual gradient mechanism for learnable clipping thresholds. The boundary gradient (Eq. 11) expands α_{clip} to preserve outliers, while the quantization gradient (Eq. 13) contracts it to minimize reconstruction error. The threshold converges when these forces reach equilibrium.

stable convergence of the learnable clipping thresholds, we employ three techniques. A global gradient norm clipping at 5.0 to prevent extreme updates from channels with occasional large outliers, $10\times$ learning rate multiplier for α_{clip} relative to LoRA parameters, allowing quantization parameters to adapt quickly during early training when fake quantization is first introduced, and a percentile-based initialization (99.5th vs max) to avoid sensitivity to extreme outliers in the calibration set. Empirically, we observe that $\alpha_{\text{clip}}^{(c)}$ converges quite early on and remains stable thereafter, with the majority of gradients flowing to the LoRA weights. This rapid convergence is consistent with prior work on learnable quantization parameters (Esser et al., 2020).

3.4 Ablation Studies

We validate our specific design choices regarding smoothing and clipping through extensive ablation studies on Llama-2-7B, evaluated on Wikitext-2

perplexity. Table 4 summarizes the results, with detailed experiment setups in Appendix A.2. Notably, we find that a 99.5th percentile based initialization for the smoothing factors outperforms the max-based initialization from SmoothQuant by 0.62 PPL, due to reduced sensitivity to extreme outliers over a limited calibration set.

4 Results

We evaluate PE-QAT on Llama-2 (7B, 13B), Llama-3-8B, and Mistral-7B-v0.3 across seven zero-shot reasoning tasks (HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018)) and WikiText-2 (Merity et al., 2016). PE-QAT was used to quantize models to W4A4KV16 with LoRA rank 32 (84.7M parameters, 1.26% of total), trained for 1500 steps on Alpaca 52K samples (Taori et al., 2023). Complete hyperparameters in Appendix A.4. As baselines, we compare against FP16, GPTQ (Frantar et al., 2023), SpinQuant-H (Liu et al., 2024), FlatQuant (Yuan et al., 2024), and L4Q (Jeon et al., 2025).

Table 1: Key results on zero-shot accuracy across all model families and benchmarks (W4A4). PE-QAT consistently achieves near-FP16 performance with minimal degradation.

Model	Method	PIQA	Wino	Hella	BoolQ	ARC-E	ARC-C	OBQA	Avg.
Llama-2-7B (6.74B parameters)									
	FP16	78.13	69.46	76.17	79.33	75.46	45.05	44.00	66.80
	SpinQuant-H[a]	77.00	66.90	73.20	74.40	72.10	47.50	54.40	66.50
	GPTQ	77.30	68.00	72.90	74.30	71.90	43.80	–	–
	FlatQuant	77.53	67.72	73.31	–	71.21	43.00	–	–
	L4Q[b]	70.80	70.20	57.20	80.40	76.90	47.10	34.80	63.62
	PE-QAT	78.78	70.24	74.75	77.28	74.46	46.93	44.40	66.69
Llama-3-8B (8.03B parameters)									
	FP16	80.47	73.32	79.29	82.11	80.60	54.01	44.60	70.62
	SpinQuant-H[a]	77.50	68.50	75.90	78.90	75.00	50.90	52.90	68.51
	FlatQuant	79.00	72.93	76.49	–	75.88	50.51	–	–
	GPTQ	78.70	70.60	76.10	78.20	75.09	48.40	–	–
	L4Q[b]	80.40	73.60	60.50	83.60	81.60	52.70	35.00	66.77
	PE-QAT	79.54	71.66	75.95	82.29	79.08	51.45	43.40	69.05
Llama-2-13B (13.0B parameters)									
	FP16	80.36	72.45	79.70	82.60	78.75	48.89	45.20	69.71
	SpinQuant-H[a]	79.50	70.80	77.50	78.10	75.90	50.80	55.20	69.68
	FlatQuant	79.65	70.56	77.88	–	76.94	48.38	–	–
	GPTQ	78.90	71.00	77.30	77.70	75.60	48.80	–	–
	L4Q[b]	80.10	71.00	60.90	82.20	79.70	51.20	35.80	65.84
	PE-QAT	79.92	71.03	78.41	81.25	76.89	49.57	44.80	68.84
Mistral-7B-v0.3 (7.25B parameters)									
	FP16	81.83	74.27	80.64	83.67	79.97	54.35	–	75.78
	SpinQuant-H[a]	80.70	71.20	78.60	80.70	76.50	53.30	–	73.50
	PE-QAT	81.88	74.59	79.34	84.16	78.83	52.82	–	75.27

[a] Requires Hadamard rotations and specialized kernels
[b] Weight-only quantization (W4A16), not full W4A4
Note: "–" indicates benchmark not reported. OBQA not evaluated for Mistral-7B.

Table 2: WikiText-2 perplexity on Llama-2-7B (W4A4). Despite higher PPL than SpinQuant-H, PE-QAT achieves superior zero-shot reasoning (Table 1), indicating better semantic preservation.

Category	Method	PPL ↓	Rel. to FP16
Reference	FP16 Baseline	5.47	1.00×
Low PPL	SpinQuant-H[a]	5.90	1.08×
	PE-QAT (Ours)	7.58	1.39x
Medium PPL	ABQ-LLM	9.31	1.70×
	SpinQuant	9.20	1.68×
	DLQAT	9.40	1.72×
	QA-LoRA	9.50	1.74×
High PPL	QLLM	11.75	2.15×
	LR-Quant	12.75	2.33×
	ZeroQAT	12.95	2.37×
	OmniQuant	14.26	2.61×
	SmoothQuant	83.12	15.19×

[a] Requires specialized kernels

5 Discussion and Future Work

PE-QAT achieves near-FP16 zero-shot accuracy (66.69 vs. 66.80 on Llama-2-7B) while training only 1.26% of parameters. We match or beat all other baselines on zero-shot tasks across model architectures. Through our extensive ablations, we demonstrate that learnable clipping thresholds (ΔPPL = +3.00 when removed, Table 4) contribute more to W4A4 adaptation than smoothing alone (ΔPPL = +0.40). We additionally show that a 99.5th percentile initialization for smoothing factors outperforms the traditional max-based approach (Appendix A.2).

WikiText-2 perplexity of 7.58 trails SpinQuant-H (5.90) but exceeds zero-shot accuracy, indicating better preservation of semantic reasoning despite elevated token-level uncertainty. KV cache remains in FP16, limiting memory savings for long-context inference. Per-channel activation quantization may face limited hardware kernel support versus per-tensor schemes; future work should validate per-tensor variants. Deployment to production INT4 kernels with end-to-end latency measurements remains future work. Integration with Hadamard rotations could close the perplexity gap to SpinQuant while maintaining parameter efficiency. Extending PE-QAT to KV cache quantization and validating per-tensor activation variants would strengthen practical deployment viability.

6 Related Work

6.1 Post-Training Quantization and Activation Outliers

Post-Training Quantization (PTQ) aims to quantize LLMs without extensive retraining, typically requiring only a small calibration dataset. Early methods like GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) focused primarily on weight quantization (e.g., W4A16), optimizing for reconstruction error. However, quantizing activations to 4-bit (W4A4) is significantly more challenging due to the presence of massive outliers in specific channels (Dettmers et al., 2022). SmoothQuant (Xiao et al., 2023) addresses this by mathematically migrating the quantization difficulty from activations to weights using a fixed smoothing factor. While effective for W8A8, fixed heuristics often fail to recover accuracy at W4A4. OmniQuant (Shao et al., 2024) improves upon this by optimizing the clipping thresholds and smoothing factors via gradient descent, yet it leaves the model weights frozen, limiting the model's ability to adapt to the quantization noise. More recently, rotation-based methods like QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2024) apply Kronecker-based rotations to flatten outlier distributions. Unlike these methods, PE-QAT jointly learns the smoothing factors and model adaptations (via LoRA), allowing the network to explicitly compensate for approximation errors during training.

6.2 Quantization-Aware Training

Quantization-Aware Training (QAT) simulates quantization noise during the forward pass to train the model to be robust to low-bit representation. LLM-QAT (Liu et al., 2023) successfully applied QAT to LLMs using data-free distillation. However, full-model QAT is computationally prohibitive, requiring memory and compute resources comparable to pre-training. This scalability bottleneck has motivated the search for parameter-efficient alternatives. Unlike Full QAT which requires backpropagating through all model weights, PE-QAT restricts gradient updates to the LoRA adapters and quantization parameters (approx. 1% of total parameters). This significantly reduces optimizer memory overhead and reduces training time compared to full model fine-tuning.

6.3 Parameter-Efficient Fine-Tuning for Quantization

Parameter-Efficient Fine-Tuning (PEFT) methods like LoRA (Hu et al., 2022) reduce training costs by freezing the base model and training low-rank adapters. QLoRA (Dettmers et al., 2024) popularized fine-tuning 4-bit quantized base models (NF4), but only quantizes weights. Recent works have attempted to bridge the gap to W4A4 using quantization-aware techniques. QA-LoRA (Xu et al., 2023) proposes group-wise quantization combined with LoRA to balance accuracy and efficiency. LoftQ (Li et al., 2023) focuses on finding a better initialization for the quantized backbone and adapters. ZeroQAT (Tan et al., 2025) employs zeroth-order optimization to perform QAT without backpropagating through the full model; while effective, zeroth-order methods can suffer from high variance in gradient estimation. L4Q (Jeon et al., 2025) is the most similar approach, also training lightweight LoRAs and quantizing merged weights. However, L4Q targets W4 quantization only (A16), while PE-QAT extends to W4A4 by jointly learning smoothing factors and activation clipping to address outliers. PE-QAT utilizes a stable, gradient-based approach with Straight-Through Estimators (STE) (Bengio et al., 2013), jointly learning SmoothQuant-style scaling factors, activation clips, and LoRA adapters. This holistic optimization allows PE-QAT to preserve near full precision accuracy on zero-shot commonsense reasoning tasks and achieve a competitive WikiText-2 perplexity at W4A4.

Limitations

The current implementation keeps the KV cache in FP16, limiting memory savings for long-context inference. Per-channel activation quantization requires specialized hardware kernels that are not yet widely supported, and production INT4 latency benchmarks have not been measured end-to-end. Experiments are limited to models up to 13B parameters; scaling behavior on larger models (e.g., 70B+) has not been validated. Additionally, evaluation focuses on zero-shot commonsense reasoning benchmarks; performance on more complex tasks such as MMLU and GSM8K is left for future work.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *Preprint*, arXiv:1308.3432.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. In *International Conference on Machine Learning*, pages 834–842. PMLR.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. 2020. Learned step size quantization. In *International Conference on Learning Representations*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Hyesung Jeon, Yulhwa Kim, and Jae-Joon Kim. 2025. L4Q: Parameter efficient quantization-aware fine-tuning on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2002–2024, Vienna, Austria. Association for Computational Linguistics.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023. Loftq: Lora-fine-tuning-aware quantization for large language models. *Preprint*, arXiv:2310.08659.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models. *Preprint*, arXiv:2305.17888.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Paul N Whatmough, Barlas Oguz, Yuhao Tang, Jinjie Zhou, Wonmin Cho, Raghuraman Krishnamoorthi, and 1 others. 2024. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Shresth Mishra, Daniel Cionca, and Raunak Pradhan. 2025. A novel framework for vehicle detection and dynamic light control via yolov8n on edge-deployed imx500. In *2025 International Conference on Machine Learning and Applications (ICMLA)*, pages 772–777.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. *Preprint*, arXiv:2308.13137.

Qitao Tan, Xiaoying Song, Jin Lu, Guoming Li, Jun Liu, Lingzi Hong, Caiwen Ding, Jundong Li, Xiaoming Zhai, Shaoyi Huang, Wei Niu, and Geng Yuan. 2025. End-to-end on-device quantization-aware training for llms at inference cost. *Preprint*, arXiv:2509.00031.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2023. Qa-lora: Quantization-aware low-rank adaptation of large language models. *Preprint*, arXiv:2309.14717.

Yuxuan Yuan, Xin Lin, Qipeng Lu, Yuxin Cai, Xin Wei, Yao Wang, Biao Song, Zheng Zhang, and Xuefei Liu. 2024. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

A Appendix

A.1 Complete Zero-Shot Results Across Models

Table 3 provides the full zero-shot accuracy results across all evaluated tasks for Llama-2-7B, Llama-2-13B, Llama-3-8B, and Mistral-7B-v0.3.

Table 3: Complete zero-shot accuracy results across all model families, benchmarks, and baseline methods. Methods marked with † require Hadamard kernels; ‡ indicates weight-only quantization.

Model	Method	PIQA	Wino	Hella	BoolQ	ARC-E	ARC-C	OBQA	Avg.
Llama-2-7B (6.74B parameters)									
	FP16 Baseline	78.13	69.46	76.17	79.33	75.46	45.05	44.00	66.80
	L4Q‡	70.80	70.20	57.20	80.40	76.90	47.10	34.80	63.62
	LR-Quant	63.71	53.19	44.42	63.39	49.96	26.27	–	–
	QLLM	67.68	56.59	58.45	–	44.40	30.89	–	–
	SpinQuant-H†	77.00	66.90	73.20	74.40	72.10	47.50	54.40	66.50
	SpinQuant	72.60	61.60	66.10	66.00	61.00	39.40	45.10	58.82
	AbQLLM	68.55	53.82	55.29	63.12	44.11	31.31	–	–
	QuaRot-GPTQ†	76.77	63.77	72.16	–	69.87	40.87	–	–
	FlatQuant	77.53	67.72	73.31	–	71.21	43.00	–	–
	RoLoRA	73.10	61.30	66.80	68.70	61.20	37.80	38.20	58.16
	DartQuant	76.93	67.17	73.76	–	70.96	44.62	–	–
	GPTQ	77.30	68.00	72.90	74.30	71.90	43.80	–	–
	MergeQuant	76.71	66.61	72.40	–	70.83	44.41	–	–
	PE-QAT (Ours)	78.78	70.24	74.75	77.28	74.46	46.93	44.40	66.69
	Δ vs FP16	*+0.65*	*+0.78*	*-1.42*	*-2.05*	*-1.00*	*+1.88*	*+0.40*	*-0.11*
Llama-3-8B (8.03B parameters)									
	FP16 Baseline	80.47	73.32	79.29	82.11	80.60	54.01	44.60	70.62
	L4Q‡	80.40	73.60	60.50	83.60	81.60	52.70	35.00	66.77
	SpinQuant-H†	77.50	68.50	75.90	78.90	75.00	50.90	52.90	68.51
	SpinQuant	68.00	59.70	59.90	53.30	56.50	35.30	37.90	52.94
	FlatQuant	79.00	72.93	76.49	–	75.88	50.51	–	–
	RoLoRA	71.10	60.20	66.70	63.20	60.30	38.20	36.80	56.64
	DartQuant	79.16	70.96	75.18	–	74.45	48.21	43.40	–
	GPTQ	78.70	70.60	76.10	78.20	75.09	48.40	–	–
	MergeQuant	77.96	69.58	74.93	–	73.70	46.38	–	–
	QUAD	77.48	68.43	75.20	–	73.74	46.50	–	–
	PE-QAT (Ours)	79.54	71.66	75.95	82.29	79.08	51.45	43.40	69.05
	Δ vs FP16	*-0.93*	*-1.66*	*-3.34*	*+0.18*	*-1.52*	*-2.56*	*-1.20*	*-1.57*
Llama-2-13B (13.0B parameters)									
	FP16 Baseline	80.36	72.45	79.70	82.60	78.75	48.89	45.20	69.71
	L4Q‡	80.10	71.00	60.90	82.20	79.70	51.20	35.80	65.84
	LR-Quant	61.70	54.85	46.32	64.19	43.06	27.56	–	–
	QLLM	70.46	55.41	62.80	–	48.48	34.39	–	–
	SpinQuant-H†	79.50	70.80	77.50	78.10	75.90	50.80	–	–
	SpinQuant	75.40	64.60	71.20	72.10	68.50	43.00	51.00	65.84
	AbQLLM	69.04	54.38	62.70	64.74	47.01	33.53	–	–
	FlatQuant	79.65	70.56	77.88	–	76.94	48.38	–	–
	RoLoRA	77.20	66.00	73.90	74.00	73.30	43.90	38.80	63.87
	DartQuant	79.27	71.11	78.04	–	75.38	47.61	44.20	–
	GPTQ	78.90	71.00	77.30	77.70	75.60	48.80	–	–
	MergeQuant	78.62	70.01	76.47	–	74.42	47.27	–	–
	PE-QAT (Ours)	79.92	71.03	78.41	81.25	76.89	49.57	44.80	68.84
	Δ vs FP16	*-0.44*	*-1.42*	*-1.29*	*-1.35*	*-1.86*	*+0.68*	*-0.40*	*-0.87*
Mistral-7B-v0.3 (7.25B parameters)									
	FP16 Baseline	81.83	74.27	80.64	83.67	79.97	54.35	–	75.78
	SpinQuant-H†	80.70	71.20	78.60	80.70	76.50	53.30	–	73.50
	SpinQuant	70.80	56.00	50.80	67.90	55.20	34.60	–	55.88
	PE-QAT (Ours)	81.88	74.59	79.34	84.16	78.83	52.82	–	75.27
	Δ vs FP16	*+0.05*	*+0.32*	*-1.30*	*+0.49*	*-1.14*	*-1.53*	–	*-0.51*

† Requires Hadamard rotations and specialized rotation kernels for deployment

‡ Weight-only quantization (W4A16), not full weight-activation W4A4

Note: "–" indicates benchmark not reported in the respective paper.

Italic rows show per-task degradation from FP16 baseline.

A.2 Ablation Study Details

In this section, we provide the full details of our ablation studies. The results are summarized in Table 4.

Table 4: Comprehensive ablation study on PE-QAT (Llama-2-7B, WikiText-2). Individual experiment details in Appendix A.2.

Ablation Category	Configuration	PPL (Δ)
Baseline	PE-QAT (Full)	7.58
Component Removal		
	w/o Learnable Clipping	10.58 (+3.00)
	w/o Activation Smoothing	7.98 (+0.40)
	w/o Quant. Gradient (PACT-only)	8.22 (+0.64)
	w/o Smoothing + Fixed Clip	12.34 (+4.76)
Smoothing Factor Initialization		
	99.5th percentile (default)	7.58
	Max (100th percentile)	8.20 (+0.62)
	Fixed at 1.0	7.95 (+0.37)
Clipping Threshold Strategy		
	99.5th pct., learnable (default)	7.58
	99.5th pct., fixed (non-learnable)	10.58 (+3.00)
Learning Rate Multiplier (Quant. Params)		
	$10\times$ (default)	7.58
	$1\times$ (uniform LR)	9.89 (+2.31)
	$5\times$	8.68 (+1.10)
	$25\times$	8.05 (+0.47)
LoRA Rank		
	$r = 32$ (default)	7.58
	$r = 16$	8.36 (+0.78)
	$r = 8$	8.50 (+0.92)

A.2.1 Smoothing Factor Initialization

We ablate the initialization of the learnable smoothing factors. In our main method, we initialize $s^{(c)}_{\text{smooth}}$ using the 99.5th percentile of activation magnitudes for channel c, which reduces sensitivity to rare activation outliers present in the calibration set. This is a deviation from the original SmoothQuant formula, so as an ablation, we instead initialize using the max (100th percentile) activation magnitude as used in the original SmoothQuant paper. Specifically, let $\mathcal{X}_c$ denote the calibration activations for channel c and $\mathcal{W}_c$ denote the weight values in column c of W_0. We define:

$$B_c = \max(|\mathcal{W}_c|)\,, \qquad A_c^{\max} = \max(|\mathcal{X}_c|)\,, \tag{9}$$

$$s^{(c)}_{\text{smooth}} = \frac{A_c^{\lambda}}{(B_c + \epsilon)^{1-\lambda}}, \quad c = 1, \ldots, k \tag{10}$$

with $\lambda = 0.5$ and $\epsilon = 10^{-6}$. We also tested a simple initialization where all smoothing factors are set to 1.0, i.e., $s^{(c)}_{\text{smooth}} = 1.0$ for all $c \in [1, k]$. As shown in Table 4, the max-based initialization degrades perplexity to 8.20 (+0.62), while fixed 1.0 initialization yields 7.95 (+0.37). The 99.5th-percentile initialization (7.58 PPL) is more robust to extreme outliers and achieves the best performance.

A.2.2 Impact of Activation Smoothing

To quantify the contribution of the learnable activation outlier smoothing, we conducted an ablation study where we disabled the smoothing component entirely. As shown in Table 4, the quantization error significantly increases, degrading the perplexity to 7.98 (+0.40). This confirms that addressing activation outliers via our learned smoothing factors is critical for maintaining performance.

A.2.3 Learning Rate Multipliers

The differentiated learning rates are critical for convergence: quantization parameters (clips and smoothing factors) adapt quickly during early training, while LoRA weights require more gradual updates to prevent overfitting. We validated the $10\times$ learning rate multiplier for quantization parameters by comparing against uniform $1\times$ multipliers, as well as $5\times$ and $25\times$ variants. As shown in Table 4, uniform $1\times$ learning rates severely degrade performance to 9.89 PPL (+2.31), confirming that quantization parameters require faster early adaptation than LoRA weights. A $5\times$ multiplier (8.68 PPL) improves over uniform but remains suboptimal, while $25\times$ (8.05 PPL) begins to destabilize training. The $10\times$ multiplier (7.58 PPL) provides optimal balance.

A.2.4 Learnable vs Fixed Activation Clipping

To validate the necessity of learning the activation clipping thresholds $\alpha^{(c)}_{\text{clip}}$, we conducted an ablation where we initialized the clipping thresholds at the 99.5th percentile (as in our main method) and then froze them. As shown in Table 4, fixed clipping thresholds degrade perplexity from 7.58 to 10.58 (+3.00), indicating that adaptive adjustment of clipping bounds is critical for minimizing quantization error. Fixed thresholds, even when well-initialized using calibration statistics, cannot account for the evolving activation distributions as LoRA adapters update during training. The learnable clipping mechanism allows $\alpha^{(c)}_{\text{clip}}$ to dynamically adjust the quantization-clipping trade-off for each channel c, expanding bounds when clipped activations carry high loss gradients and contracting them when quantization error dominates.

A.2.5 Activation Clip Gradient Components

To validate the contribution of our quantization error gradient term, we compare the full PE-QAT gradient formulation ((14)) against a symmetric PACT-style (Choi et al., 2018) boundary-only variant that omits the LSQ-inspired quantization component. Specifically, we disable the quantization scale gradient ((13)), retaining only the bound-

ary gradient ((11)): As shown in Table 4, the boundary-only configuration increases perplexity from 7.58 to 8.22 (+0.64). This demonstrates that the boundary gradient alone signals when to expand clipping thresholds but provides no feedback about discretization quality within the clipping range. For aggressive quantization with only 16 discrete levels per channel (with $Q_{\max} = 7$ for 4-bit), explicit minimization of reconstruction error $\epsilon^{(t,c)} = \tilde{X}^{(t,c)} - X^{(t,c)}_{\text{clipped}}$ via gradient descent is critical.

A.2.6 LoRA Rank Selection

We ablate the LoRA rank r to assess its impact on WikiText-2 perplexity. While higher ranks provide greater model capacity for compensating quantization error, they also increase trainable parameters and risk overfitting. We trained PE-QAT with $r = 8$, $r = 16$, and $r = 32$ while keeping all other hyperparameters identical. As shown in Table 4, reducing rank from 32 to 16 degrades perplexity from 7.58 to 8.36 (+0.78), while rank 8 further degrades to 8.50 (+0.92). This indicates that $r = 32$ provides a favorable balance between parameter efficiency and adaptation capacity for W4A4 quantization. The relatively modest degradation suggests that further rank reduction could be viable for extremely resource-constrained scenarios, though we maintain $r = 32$ for optimal performance.

A.3 Gradient Derivation for Learnable Clipping Thresholds

Intuitively, the gradient must balance two objectives: preserving outlier information by expanding clipping bounds when large activations are truncated, and improving discretization quality by adjusting quantization bin size within the clipping range. The clipping operation is non-differentiable, requiring a custom backward pass. We derive gradients for the per-channel clipping thresholds $\alpha^{(c)}_{\text{clip}}$ by decomposing the loss contribution into two components, inspired by PACT (Choi et al., 2018) and LSQ (Esser et al., 2020). Let $X'^{(t,c)}$ denote the smoothed activation (input to clipping), $X^{(t,c)}_{\text{clipped}} = \text{clip}(X'^{(t,c)}, -\alpha^{(c)}_{\text{clip}}, \alpha^{(c)}_{\text{clip}})$ the clipped activation, and $\tilde{X}^{(t,c)}$ the fake-quantized (dequantized) activation. Component 1 is the boundary gradient. Values exceeding the clipping bounds contribute directly to the gradient. For symmetric clipping at $\pm\alpha^{(c)}_{\text{clip}}$ for channel c, we check the smoothed activations $X'^{(t,c)}$ against the threshold and sum over all N tokens (batch size $\times$ sequence length):

$$\frac{\partial L}{\partial \alpha^{(c)}_{\text{clip}}}^{(\text{bound})} = \sum_{t=1}^{N} \left[\mathbb{1}_{X'^{(t,c)} > \alpha^{(c)}_{\text{clip}}} \cdot \frac{\partial L}{\partial Y^{(t)}} - \mathbb{1}_{X'^{(t,c)} < -\alpha^{(c)}_{\text{clip}}} \cdot \frac{\partial L}{\partial Y^{(t)}} \right] \tag{11}$$

where t indexes tokens, c indexes channels, $Y^{(t)}$ is the corresponding output element, and $\mathbb{1}$ is the indicator function. This term encourages $\alpha^{(c)}_{\text{clip}}$ to expand when smoothed activations that would be clipped have large loss gradients, reducing information loss from hard clipping. Component 2 is the quantization scale gradient. For smoothed activations within the clipping range $|X'^{(t,c)}| \leq \alpha^{(c)}_{\text{clip}}$, no clipping occurs and we have $X^{(t,c)}_{\text{clipped}} = X'^{(t,c)}$. The quantization scale $s^{(c)}_x = \alpha^{(c)}_{\text{clip}} / Q_{\max}$ (where $Q_{\max} = 2^{b-1} - 1 = 7$ for $b = 4$) determines discretization granularity. The per-element reconstruction error is computed by comparing the fake-quantized activation to the clipped activation:

$$\begin{aligned} \epsilon^{(t,c)} &= \tilde{X}^{(t,c)} - X^{(t,c)}_{\text{clipped}} \\ &= \text{round}\left(\frac{X^{(t,c)}_{\text{clipped}}}{s^{(c)}_x} \right) \cdot s^{(c)}_x - X^{(t,c)}_{\text{clipped}} \end{aligned} \tag{12}$$

where round$(\cdot)$ denotes round-to-nearest-integer. Following LSQ (Esser et al., 2020), we compute the gradient with respect to the clipping threshold, summing only over tokens where the smoothed activation falls within the clipping range:

$$\frac{\partial L}{\partial \alpha^{(c)}_{\text{clip}}}^{(\text{quant})} = \sum_{t=1}^{N} \mathbb{1}_{|X'^{(t,c)}| \leq \alpha^{(c)}_{\text{clip}}} \cdot \frac{\partial L}{\partial Y^{(t)}} \cdot \frac{\epsilon^{(t,c)}}{\max\left(\alpha^{(c)}_{\text{clip}}, \epsilon_{\text{safe}}\right)} \tag{13}$$

where $\epsilon_{\text{safe}} = 0.1$ prevents division by very small clipping values during gradient computation. This term adjusts $\alpha^{(c)}_{\text{clip}}$ to minimize quantization error for activations within the clipping range. Increasing $\alpha^{(c)}_{\text{clip}}$ enlarges quantization bins (reducing rounding error) but decreases resolution (increasing approximation error). For the rounding operation round$(\cdot)$,

we use the standard STE (Bengio et al., 2013), passing gradients through as if $\text{round}(z) = z$ during backpropagation. The total gradient for channel c combines both components:

$$\frac{\partial L}{\partial \alpha_{\text{clip}}^{(c)}} = \frac{\partial L}{\partial \alpha_{\text{clip}}^{(c)}}^{(\text{bound})} + \frac{\partial L}{\partial \alpha_{\text{clip}}^{(c)}}^{(\text{quant})} \tag{14}$$

The gradient in (14) balances two competing objectives. When large smoothed activations exceed the clipping bounds, $\frac{\partial L}{\partial \alpha_{\text{clip}}^{(c)}}^{(\text{bound})} > 0$ pushes $\alpha_{\text{clip}}^{(c)}$ upward to preserve signal. For activations within the clipping range, when quantization error dominates, $\frac{\partial L}{\partial \alpha_{\text{clip}}^{(c)}}^{(\text{quant})}$ can be negative if finer granularity (smaller $\alpha_{\text{clip}}^{(c)}$) reduces discretization noise. This creates a natural equilibrium where $\alpha_{\text{clip}}^{(c)}$ converges to a value that minimizes the combined loss from both clipping and quantization. However, the gradient magnitude can vary significantly across channels and training phases. To ensure stable convergence, we employ three regularization techniques: global gradient norm clipping at 5.0 to prevent extreme updates from channels with occasional large outliers; a $10\times$ learning rate multiplier for α_{clip} compared to LoRA parameters, allowing quantization parameters to adapt quickly during early training when fake quantization is first introduced while preventing interference with slower LoRA convergence; and initialization at the 99.5th percentile (rather than max), providing a stable starting point that is robust to extreme outliers in the calibration set and reducing the risk of divergence. Empirically, we observe that α_{clip} converges relatively early after which the bulk of the gradients flowing to the LoRA weights. This rapid convergence is consistent with prior work on learnable quantization parameters (Esser et al., 2020).

A.4 Complete Training Hyperparameters

Table 5 provides the complete set of hyperparameters used for PE-QAT training.

Table 5: Hyperparameters for PE-QAT training on Llama-2-7B.

Component	Configuration
Model & Quantization	
Base Model	Llama-2-7B
Weight/Activation Bits	4-bit / 4-bit (per-channel symmetric)
KV Cache	16-bit (unquantized)
Sequence Length	2048
LoRA Configuration	
Rank (r)	32
Alpha (α)	64
Dropout	0.05
Target Modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
Calibration	
Calibration Steps	200
Activation Clip Percentile	99.5%
Smoothing λ	0.5
Calibration Samples	128
Training	
Dataset	Alpaca (52K instruction-response pairs)
Training Steps	1500
Per-Device Batch Size	16
Gradient Accumulation Steps	2 (effective batch size = 32)
Base Learning Rate	1×10^{-4}
LR Schedule	Cosine with 15% warmup, min LR = 10% peak
Learning Rate Multipliers	LoRA: 1.0×, Clips: 10.0×, Smooth: 10.0×
Optimizer	AdamW (β_1=0.9, β_2=0.95, ϵ=10^{-8})
Weight Decay	0.01 (LoRA only)
Gradient Clipping	Max norm = 5.0
Gradient Checkpointing	Enabled (non-reentrant)
System	
Precision	bfloat16 (TF32 enabled)
Hardware	NVIDIA GH200 Chip
Random Seed	42