QUAD: Quantization and Parameter-Efficient Fine-Tuning of LLM with Activation Decomposition

Anonymous ACL submission

Abstract

Large Language Models (LLMs) excel in diverse applications but suffer inefficiency due to massive scale. While quantization reduces computational costs, existing methods degrade accuracy in medium-sized LLMs (e.g., Llama-3-8B) due to activation outliers. To address this, we propose QUAD (Quantization with Activation Decomposition), a framework leveraging Singular Value Decomposition (SVD) and Householder transformation to suppress activation outliers for effective 4-bit quantization. QUAD estimates activation singular vectors offline using calibration data to construct a transformation matrix P, shifting outliers to additional dimensions in full precision while quantizing the remaining components to 4-bit. Additionally, QUAD enables parameter-efficient fine-tuning via adaptable full-precision outlier weights, narrowing the accuracy gap between quantized and full-precision models. Experiments demonstrate that QUAD achieves 95–97% accuracy under W4A4 quantization and parameter-efficient fine-tuning for Llama-3 and Qwen-2.5 models. Our code is available at our repository.

1 Introduction

Large Language Models (LLMs) (Grattafiori et al., 2024; Yang et al., 2024; Liu et al., 2024a) have demonstrated remarkable performance across numerous fields and have been widely adopted in various applications, such as chat assistants, coding copilots (Roziere et al., 2023; Hui et al., 2024; Li et al., 2024a), and beyond. However, the scaling law (Kaplan et al., 2020) has led to increasingly deeper LLMs with hundreds of billions of parameters, rendering them inefficient for text-processing tasks. Concurrently, existing work (Zhu et al., 2024) highlights that for throughput-oriented serving systems, many workloads are predominantly compute-bound. Consequently, there is a critical need for effective methods to compress LLMs and enhance the efficiency of General Matrix Multiplication (GEMM) operations, given that GEMM dominates computational tasks within LLMs.

Weight and activation quantization aims to address this issue by representing both weights and activations in lower precision, thereby reducing storage overhead and leveraging more efficient hardware, such as INT4 tensor cores, to accelerate GEMM computations. While existing approaches (Frantar et al., 2022; Lin et al., 2024b; Du et al., 2024) have successfully quantized LLM weights to 4 bits or even lower with nearly no loss in accuracy, quantizing both weights and activations remains challenging. This difficulty arises due to the higher prevalence of outliers in activations, making them harder to quantify than weights. Existing methods like QuaRot (Ashkboos et al., 2024b), SpinQuant (Liu et al., 2024c), and FlatQuant (Sun et al., 2025) attempt to suppress outliers in both weights and activations by rotating the matrix and quantizing both to 4 bits. These techniques perform well for large-scale LLMs, such as Llama-2-70B, yet still result in significant performance degradation for medium-sized LLMs, such as Llama-3-8B. We attribute this to the fact that for medium-sized LLMs, the rotation matrix is insufficient in suppressing outliers, making it difficult to quantize activations to 4 bits. Thus, more effective methods are required to eliminate outliers from activations.

Singular Value Decomposition (SVD) is an effective tool used to decompose a matrix's high and low-frequency components, thus enabling the removal of outliers from the matrix (Li et al., 2024b). Leveraging the impressive performance of SVD in eliminating outliers, we apply it to activations and propose our method, named **QUAD** (**Qu**antization with **A**ctivation **D**ecomposition). However, migrating SVD to activations presents several challenges. **Firstly**, weights remain static, whereas activations dynamically change with varying in-

puts during serving. **Secondly**, while the SVD of weight matrices can be precomputed offline without time constraints, applying SVD to dynamically generated activations during inference imposes a prohibitive computational overhead. **Thirdly**, the SVD method should be compatible with existing rotation methods. To address these challenges, QUAD estimates the singular vectors of activations (Ashkboos et al., 2024a; Hu et al., 2024) offline using a small amount of calibration data. It then constructs the transformation matrix $P \in \mathbb{R}^{(C+r)\times C}$ based on these singular vectors and Householder transformation (Householder, 1958) ($H_u(x) = x - 2\langle x \cdot u\rangle \cdot u$). The matrix P has two key properties: (1) it shifts outliers in activations to an additional r dimension, thereby eliminating outliers in the original activations; (2) it satisfies $P^\top P = I$. Consequently, matrix P enables equivalent transformations of the LLM and is compatible with existing methods. After transformation, for GEMM computations, most weights and activations are quantized to 4 bits, while the r outlier dimensions and their corresponding weights are retained in full precision.

In addition to removing activation outliers, QUAD can also be applied to parameter-efficient fine-tuning of quantized models (Hu et al., 2022; Dettmers et al., 2023; Huang et al., 2024). Specifically, we retain the weights corresponding to the outlier dimensions in full precision, meaning these portions of the weights can serve as parameter-efficient adapters to fine-tune the model. Meanwhile, we demonstrate that the adapter initialized by QUAD provides a sub-optimal solution for approximating full fine-tuning with unchanged input distributions. Based on the full-precision portion of the model, we further reduce the gap between the quantized and full-precision models through parameter-efficient fine-tuning.

To evaluate the effectiveness of QUAD, we conducted extensive experiments on diverse LLMs and datasets. The contributions of this work are summarized as follows:

- We introduce QUAD, which suppresses outliers in activations via SVD decomposition and integrates with existing rotation methods to enhance the performance of quantized models.
- Beyond quantization, QUAD can also be used for parameter-efficient fine-tuning, and we propose fine-tuning the full-precision portion of the quantized model to further bridge the gap between the quantized and full-precision models.
- Based on these improvements, QUAD maintains 95-97% of the full-accuracy model's performance with W4A4 quantization and parameter-efficient fine-tuning.

2 Related Work

Quantization. Quantization is a commonly used approach for LLM deployment to compress storage space and improve inference speed by representing weights and activations in lower bits. Existing quantization methods can be broadly classified into two categories: Quantization-aware training (QAT) and Post-training quantization (PTQ). QAT combines quantization with training and fine-tuning to represent the model with lower precision while maintaining model performance. Representative QAT methods include LLM-QAT (Liu et al., 2023), BitDistiller (Du et al., 2024), EfficientQAT (Chen et al., 2024), and the BitNet series (Wang et al., 2023, 2024a). Since the QAT method has large resources and time overhead, more work focuses on PTQ, which can achieve quantization with only a small amount of calibration data. The main challenge of PTQ methods comes from the outliers in the parameters and activations of LLM, which bring large quantization errors. Therefore, existing methods have proposed the following methods to overcome the impact of outliers, including model equivalent transformation and weight compensation. Equivalent transformations of the model include shifting, scaling, and rotating. SmoothQuant (Xiao et al., 2023) and AWQ (Lin et al., 2024b) propose employing scaling operations, while OS+ (Wei et al., 2022, 2023) proposes shifting operations to suppress outliers. In addition to scaling and shifting, QUIP (Chee et al., 2023), QuaRot (Ashkboos et al., 2024b), DuQuant (Lin et al., 2024a), and SpinQuant (Liu et al., 2024c) further utilize rotation operations to suppress outliers. These transformations are also used in subsequent work, such as QUIK (Ashkboos et al., 2023), QmniQuant (Shao et al., 2023), AffineQuant (Ma et al., 2024), QServe (Lin et al., 2024c), and OSTQuant (Hu et al., 2025). The weight compensation technique, which improves quantization by adjusting the weights during the quantization process, was first proposed by OBS (Hassibi et al.,

1993) and subsequently widely applied to LLM by GPTQ (Frantar et al., 2022). In addition to transformations and compensation, existing work also uses mixed-precision (Dettmers et al., 2022) and non-uniform data types (Van Baalen et al., 2024; Liu et al., 2024b) to improve quantization. Recent work ResQ (Saxena et al., 2025) shares a similar motivation with our approach; however, it employs a different matrix transformation. Moreover, our method incorporates parameter-efficient fine-tuning and seamlessly integrates it with existing techniques.

Parameter-Efficient Fine-Tuning. Parameter-efficient fine-tuning (PEFT) is widely employed to fine-tune models under resource-constrained scenarios while maintaining performance close to full fine-tuning. Among these methods, LoRA (Hu et al., 2022) and its series of variants (Wang et al., 2024b, 2025b; Meng et al., 2025) are the most commonly adopted PEFT approaches. These techniques preserve the performance of fine-tuning while reducing the computational overhead by updating the model within a low-rank subspace. Moreover, beyond their application to full-precision models, existing research has extended LoRA to the fine-tuning of quantized models. Notably, QLoRA (Dettmers et al., 2023), INT2.1 (Chai et al., 2023), LQ-LoRA (Guo et al., 2024), LoftQ (Li et al., 2023), PEQA (Kim et al., 2023), and QA-LoRA (Xu et al., 2023) leverage LoRA for weight-only quantized models, while RoLoRA (Huang et al., 2024) further incorporates LoRA for both weight and activation quantized models.

3 Background

3.1 Transformer Architecture

A large language model (LLM) typically comprises an embedding layer, a sequence of transformer layers, and a language model head. Figure 1 depicts the architecture of a standard transformer layer, which consists of three core components: the RMSNorm module, the Attention module, and the Feed-Forward Network (FFN) module. These layers collectively involve seven weight matrices: W_Q, W_K, W_V, W_O, W_{up}, W_{gate}, and W_{down}. To analyze their roles systematically, we categorize these matrices into two groups based on their position relative to module inputs and outputs. Specifically, **U-type matrices**—those positioned near module inputs—include W_Q, W_K, W_V, W_{up}, and W_{gate}, while **D-type matrices**—situated near outputs—comprise W_O and W_{down}.

3.2 Equivalence Transformations

Prior research demonstrates that an LLM's output remains invariant under certain linear transformations of its weight matrices. A key technique involves applying an orthogonal matrix Q ($Q^\top Q = I$) to perform equivalence transformations. For U-type matrices, this entails left-multiplying by Q, followed by right-multiplying their corresponding D-type matrices by $Q^\top$ to preserve the overall computation. This approach remains valid even in the presence of RMSNorm layers between modules, as shown by the following identity:

$$\text{RMSNorm}(X) = \text{RMSNorm}(XQ^\top)Q. \quad (1)$$

Practically, we first absorb the scaling parameters of adjacent RMSNorm modules into neighboring weight matrices as described in prior work like QuaRot. After this absorption, the RMSNorm operation simplifies to $\text{RMSNorm}(x) = x/\|x\|$. Then, we can transform the weights by rotating the matrix, $W_U \leftarrow QW_U$, $W_D \leftarrow W_D Q^\top$, where W_U, W_D denote the U-type and D-type weight matrices. It should be noted that the equivalence transformation is not limited to rotation matrices; any matrix Q (satisfying $Q^\top Q = I$) can be used to transform the weights of the model while preserving its output unchanged. The proof is shown in Appendix C.1.

4 Method

Our proposed method, QUAD, comprises three stages: transformation (Section 4.1), quantization (Section 4.2), and parameter-efficient tuning (Section 4.3). In the transformation stage, we utilize singular vectors of activations and Householder transformation to construct a projection matrix for mapping outliers in the activations to additional dimensions. Subsequently, weight matrices and activations are smoothed using a rotation matrix. Following the approach of QuaRot, in the quantization phase, we apply GPTQ to quantize the weight matrix and inject quantization operators into the model to quantify activations online using the round-to-nearest method. Notably, the weights and activations corresponding to the outlier dimensions retain full precision. Finally, in the parameter-efficient tuning stage, we fine-tune the full-precision part of the quantized model using high-quality data.

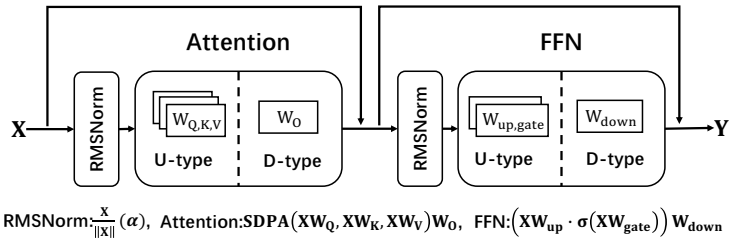

Figure 1: Example of a transformer layer structure.

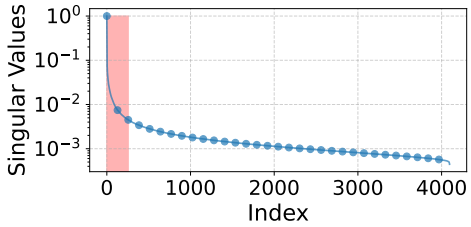

Figure 2: Estimated singular values.

4.1 Transformation

Model equivalence transformation aims to project outliers in activations into additional dimensions and smooth the original weight matrices and activations. To achieve this, we first fuse the scaling parameters (α) of each RMSNorm module into adjacent weight matrices. Next, we construct a projection matrix based on calibration data and singular value decomposition (SVD). Let $X^{(i)} \in \mathbb{R}^{B\times h}$ denote the input activation of layer i of the model, where B represents the number of tokens in the calibration data, and h represents the dimensions of the activation. We can estimate the singular vectors of the activation using the following formula:

$$U, \Sigma, U^\top = \text{SVD}(\sum_i X^{(i)\top} X^{(i)}). \qquad (2)$$

Here, the columns of U correspond to the estimated singular vectors of the activation, and Σ contains the corresponding singular values. Figure 2 shows the magnitude of the singular values for different singular vectors, revealing that a small subset of singular vectors dominates the singular values. We hypothesize that these dominant singular vectors contribute to the presence of outliers in the activation. To mitigate this, we propose removing the components associated with these dominant singular vectors from the existing activations via Householder transformation and projecting them into additional dimensions. Specifically, if the first r singular vectors are to be removed, the projection matrix can be constructed as follows:

$$P = \left(U_{1:r}, I - \textstyle\sum_{i=1}^{r} U_i U_i^\top\right) \in \mathbb{R}^{h\times(r+h)} \qquad (3)$$

It can be proven that the matrix P satisfies $PP^\top = I$ (see Appendix C.2), making it suitable for model equivalence transformation. Furthermore, by letting $\hat{X} = XP$, the outliers in X can be projected into the first r dimensions of $\hat{X}$. Subsequently, we employ a random Hadamard matrix H to construct a rotation matrix Q, which further smooths the weights and activations.

$$Q = \begin{pmatrix} H_{r\times r} & 0 \\ 0 & H_{h\times h} \end{pmatrix} \in \mathbb{R}^{(r+h)\times(r+h)}. \qquad (4)$$

Here, the Hadamard matrix is a specialized rotation matrix that can be utilized to smooth weights and activations and can be computed efficiently. Please refer to Appendix B for a more detailed explanation of the Hadamard matrix.

Finally, the introduced projection and rotation matrices are fused with the existing weight matrices. For a U-type weight matrix W_U and a D-type weight matrix W_D, they are updated as follows:

$$W_U \leftarrow Q^\top P^\top(\alpha) W_U, \quad W_D \leftarrow W_D P Q.$$

Figure 3 illustrates the FFN module of the transformed model, and the Attention module of the transformed model is shown in Appendix A. By combining the projection and rotation, we smooth the matrix and activations and project the outliers into additional r dimensions. Additionally, following QuaRot and SpinQuant, we add online Hadamard operators before W_O and W_{down} to smooth the intermediate activations of the Attention module and the FFN module. Thus, for a D-type matrix W_D, we further multiply the Hadamard matrix H on its left-hand side:

$$W_D \leftarrow H W_D P Q.$$

4.2 Quantization

To enhance the computational efficiency of GEMM, we quantize both weights and activations using a symmetric per-row (per-token) approach. Specifically, after applying transformations, we use GPTQ to quantize the weights of large language models (LLMs). Before each linear layer, we quantize activations online using the round-to-nearest (RTN) method. This process involves dividing the maximum absolute value of each token by the highest

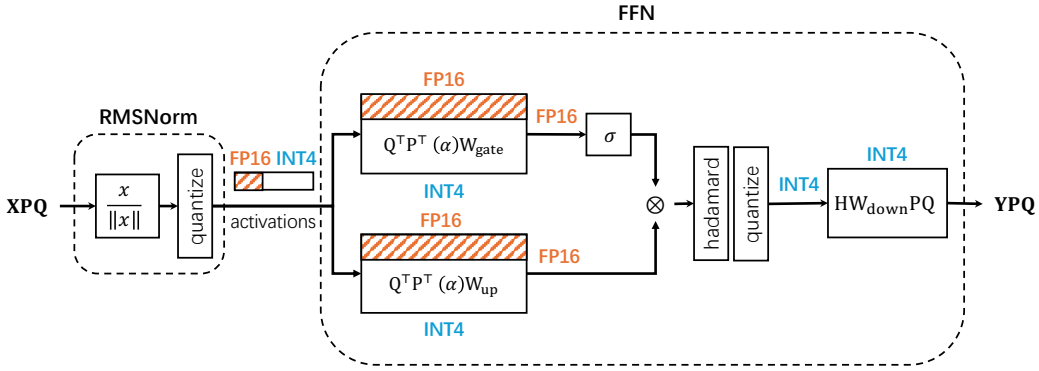

Figure 3: The FFN module of the transformer layer after applying QUAD transformation.

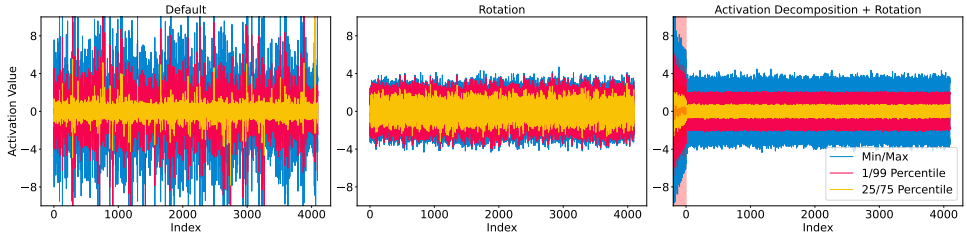

Figure 4: The input activations of the transformer layer under three scenarios: default, post-rotation, and post-decomposition with post-rotation. It can be seen that activation decomposition can effectively reduce the magnitude of the 1/99th percentile, thus making the activation easier to quantize.

value expressible with the target quantization precision to determine the scale for each token. Each token is then divided by its corresponding scale and rounded to the nearest integer.

Since we introduce an extra r dimension for the activations before the U-type linear layer to represent outliers, the weights and activations corresponding to this component are kept in full precision, while the remaining components are quantized to reduce performance degradation due to quantization.

4.3 Quantization-Aware Parameter-Efficient Fine-Tuning

The additional weights corresponding to outliers can also be leveraged for efficient model parameter tuning. Thus, we aim to minimize the gap between the quantized and full-precision models through fine-tuning with high-quality datasets. Fine-tuning is applied to the full-precision sub-matrix associated with outlier dimensions for U-type layers. Meanwhile, following EfficientQAT (Chen et al., 2024), we also adjust the scale corresponding to the quantization matrix for both U-type and D-type weight matrices. We adopt the straight-through estimator (Bengio et al., 2013) (STE) for gradient transmission to handle non-differentiable operations, such as quantization during model training, i.e., $\nabla X \leftarrow \nabla X_q$, where $X_q = \text{Dequantize}(\text{Quantize}(X))$.

4.4 Theoretical Analyses

Quantization

Assume the input to a linear layer is $X \in \mathbb{R}^{B\times m}$ and the weight matrix is $W \in \mathbb{R}^{m\times n}$. Following SVDQuant (Li et al., 2024b), we define the quantization error as $E(X,W) = \|XW - Q(X)Q(W)\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm, and further present the following propositions:

Proposition 4.1. *The quantization error can be decomposed as follows:*

$$E(X,W) \leq \|X\|_F\|W - Q(W)\|_F + \|X - Q(X)\|_F\|Q(W)\|_F.$$

Proposition 4.2. *For the round-to-nearest (RTN) quantization method Q, if the tensor X follows a normal distribution, we have:*

$$\mathbb{E}\left[\|X - Q(X)\|_F\right] \leq \frac{\sqrt{\log(size(X)\pi)}}{q_{max}}\mathbb{E}\left[\|X\|_F\right],$$

Our core idea is to reduce the difficulty of quantifying the input by projecting outliers to additional dimensions. Specifically, we decompose the inputs and weights into two parts: $\hat{X}, \hat{W}$ and X_r, W_r, respectively, such that $XW = \hat{X}\hat{W} + X_r W_r$. Therefore, we can rewrite the quantization error as follows:

$$\begin{aligned} E(X, W) &= \|XW - X_r W_r - Q(\hat{X})Q(\hat{W})\|_F \\ &= \|\hat{X}\hat{W} - Q(\hat{X})Q(\hat{W})\|_F. \end{aligned}$$

From the propositions above, it is evident that the quantization error correlates with both the magnitude of the inputs and their quantization errors. Moreover, the quantization error is further unified with the input magnitude, as the quantization error of the input is inherently limited by its magnitude. Consequently, reducing the input X's magnitude emerges as a viable strategy to minimize the quantization error. Since $\|X\|_F = \sqrt{\sum_{i=1}^{\min(n,m)} \sigma_i^2}$, our approach involves minimizing the input's norm by eliminating the largest r singular values. This process is formalized as $X_r = X\left(U_{1:r}\right)$ and $\hat{X} = X\left(I - \sum_{i=1}^{r} U_i U_i^\top\right)$, where $U, \Sigma, U^\top = \text{SVD}(X^\top X)$.

Parameter-Efficient Fine-Tuning

In addition to reducing the quantization error, the full-precision part of the weight matrix W_r can also be used for parameter-efficient fine-tuning of the quantized model.

Proposition 4.3. *Parameter-efficient tuning via $W_r = U_{1:r}^\top W$ is a suboptimal solution to approximate full fine-tuning with unchanged input distribution. Specifically, suppose the gradients of the weights W and W_r are ∇W and ∇W_r, then we have:*

$$X\nabla W \approx X_r \nabla W_r.$$

Consider the output of the linear layer is $Y = XW$, and the gradient of Y is ∇Y. Given $\nabla W = X^\top \nabla Y$, then based on the weight gradients and inputs, we can estimate the variation of the linear layer outputs as $\Delta Y = X\nabla W = XX^\top \nabla Y$. Similarly, we have $\Delta Y = X_r \nabla W_r = X_r X_r^\top \nabla Y$ for the quantized linear layer. Thus, we can approximate $XX^\top$ with $X_r X_r^\top$ such that the parameter-efficient fine-tuning approximates the full fine-tuning, where the optimal solution is $X_r = X(U_{1:r})$, i.e., $W_r = (U_{1:r})^\top W$, where $U, \Sigma, U^\top = \text{SVD}(X^\top X)$. The proof of the above propositions is given in Appendix C.

5 Experiment

5.1 Experimental Setup

Models, Datasets, and Baselines. We evaluate the performance of QUAD on the Llama-2 (Touvron et al., 2023), Llama-3 (Grattafiori et al., 2024), and Qwen-2.5 (Yang et al., 2024) model families across zero-shot and generation tasks using the LM-evaluation-harness framework under default parameter configurations. For zero-shot evaluation, we utilize the benchmark datasets PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019), HellaSwag (Zellers et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), and LAMBADA (Paperno et al., 2016). For generation tasks, we employ GSM8K (Cobbe et al., 2021) and HumanEval (Chen et al., 2021). QUAD is compared against the post-training quantization (PTQ) method QuaRot (Ashkboos et al., 2024b). Considering that on top of QuaRot, existing work also uses more sophisticated rotation and scaling methods to improve the effectiveness of the quantization method, for this reason, we also combine QUAD with the existing SOTA quantization method, FlatQuant, and compare it with SpinQuant (Liu et al., 2024c), OstQuant (Hu et al., 2025), and FlatQuant (Sun et al., 2025) itself.

Implementation Details. QUAD is implemented using PyTorch (Paszke et al., 2019) and the Hugging Face Transformers (Wolf et al., 2020) library. During activation decomposition, the top 64 singular vectors ($r = 64$) are projected onto an additional dimension. Weight quantization is performed via GPTQ (Frantar et al., 2022), where the clipping ratio is determined through a linear search over squared error metrics. Activation quantization adopts a round-to-nearest method with a fixed clipping ratio of 0.9. Symmetric quantization is applied to weights and activations: per-channel quantization for weights and per-token quantization for activations. Custom CUDA kernels via Tilelang (Wang et al., 2025a) are developed for the quantization/dequantization of activations, and leveraging the CUTLASS library (Thakkar et al., 2023) to accelerate GEMM execution. Calibration datasets consist of 128 samples from the C4 dataset (Raffel et al., 2023) for base models and 128 samples from Meta-Math-QA (Yu et al., 2024) and Code-Feedback (Zheng et al., 2025) for instruction-tuned models, each with a sequence length of 2048. During the parameter-efficient fine-tuning stage,

Model	Method	PQ	WG	HS	A-e	A-c	LA	Avg.
Llama-3.2-3B	Baseline	77.48	69.93	73.60	71.68	45.90	70.50	68.18
	QuaRot	74.05	63.30	66.91	63.55	40.27	61.50	61.60
	QUAD	74.65	65.75	70.01	65.11	42.41	**65.48**	63.90
	(w. train)	**76.10**	**67.20**	**70.72**	**67.15**	**42.94**	64.39	**64.75**
Llama-3-8B	Baseline	80.79	73.16	79.16	77.74	53.24	75.74	73.30
	QuaRot	75.30	66.46	72.56	67.89	42.41	67.94	65.43
	QUAD	77.48	68.43	75.22	73.74	46.50	**71.18**	68.76
	(w. train)	**78.18**	**68.63**	**76.43**	**75.38**	**48.50**	70.18	**69.55**

Table 1: Zero-shot accuracy of LLAMA-3 models with W4A4 quantization.

Model	Method	PQ	WG	HS	A-e	A-c	LA	Avg.
Qwen2.5-1.5B	Baseline	75.95	63.46	67.73	71.68	45.48	62.06	64.39
	QuaRot	70.13	56.35	59.78	64.06	37.03	47.51	55.81
	QUAD	72.36	59.19	62.70	**68.81**	**41.55**	56.01	60.10
	(w. train)	**73.45**	**59.72**	**65.12**	68.00	40.54	**56.05**	**60.48**
Qwen2.5-3B	Baseline	78.40	68.51	73.59	72.98	46.93	66.93	67.89
	QuaRot	73.83	61.33	65.80	65.53	40.10	51.14	59.62
	QUAD	75.57	**62.67**	69.14	70.12	44.03	60.45	63.66
	(w. train)	**76.81**	61.70	**70.67**	**72.81**	**46.55**	**60.57**	**64.85**
Qwen2.5-7B	Baseline	79.98	72.69	78.90	77.31	51.11	71.71	71.95
	QuaRot	76.55	65.59	74.59	**75.00**	47.95	65.44	67.52
	QUAD	78.89	**68.67**	75.66	74.75	48.46	67.59	69.00
	(w. train)	**79.36**	67.62	**76.83**	**77.40**	**50.87**	**68.04**	**70.02**

Table 2: Zero-shot accuracy of Qwen-2.5 models with W4A4 quantization.

Model	Method	GSM8K	HumanEval@1
Qwen2.5-3B-Instruct	Baseline	50.23	48.17
	QuaRot	**37.76**	20.12
	QUAD	36.16	**29.27**
Qwen2.5-7B-Instruct	Baseline	74.75	64.02
	QuaRot	64.44	50.00
	QUAD	**68.61**	**51.22**

Table 3: Generation tasks performance of Qwen-2.5-Instruct models

Model	Method	GSM8K	HumanEval@1
Qwen2.5-3B	QLoRA	57.13	37.20
	QUAD	**57.70**	**38.41**
Qwen2.5-7B	QLoRA	71.58	54.27
	QUAD	**72.25**	**56.10**

Table 4: Generation tasks performance of Qwen-2.5 models after parameter-efficient tuning under W4A16 quantization

we employed the Alpaca dataset (Taori et al., 2023) for base models and utilized the training sets from Meta-Math-QA and Code-Feedback for instruction-tuned models. We adopted the AdamW optimizer with a learning rate of 1e-5. All experiments were conducted on a system equipped with four NVIDIA A6000 (48GB) GPUs.

5.2 Experimental Results

Zero-Shot Task Performance. Table 1 and Table 2 compare QUAD and baseline methods on zero-shot accuracy for Llama-3 and Qwen-2 models. It can be seen that QUAD outperforms QuaRot with W4A4 quantization, achieving 94.9% accuracy for Llama-3-8B and 97.3% for Qwen-2.5-7B.

In addition to QuaRot, we also combined QUAD

Model	Method	PQ	WG	HS	A-e	A-c	LA	Avg.
Llama-3-8B	Baseline	80.63	73.01	79.18	77.74	53.50	75.74	73.30
	SpinQuant	79.16	68.75	75.43	73.57	46.33	71.40	69.11
	OSTQuant	78.51	69.22	76.01	**76.73**	49.32	70.77	70.09
	FlatQuant	79.38	71.03	76.35	75.67	49.49	72.71	70.77
	FlatQUAD	**79.65**	**71.11**	**76.93**	74.71	**50.17**	**74.13**	**71.12**
	(w. train)	78.35	73.01	77.10	77.36	50.34	74.27	71.74

Table 5: Comparison of QUAD and advanced rotation-based methods on LLAMA models with W4A4 quantization.

with FlatQuant (FlatQUAD), and compared it with SpinQuant (Liu et al., 2024c), OSTQuant (Hu et al., 2025), and FlatQuant (Sun et al., 2025) itself. The experimental results are presented in Table 5. It is evident that the FlatQUAD surpasses existing methods and achieves 97.9% accuracy for Llama-3-8B. Experiment results for more models (e.g., Llama-2), baselines (e.g., AWQ, SmoothQuant, GPTQ, and OmniQuant), and mix-precisions (W4A4 and W4A4/A8) are provided in Appendix D.

Generation Task Performance. Table 3 evaluates QUAD on generation tasks for the Qwen-2-Instruct family. At W4A4 precision, QUAD surpasses QuaRot by 4.17% on GSM8K and 1.22% on HumanEval for Qwen-2.5-7B-Instruct.

Parameter-Efficient Fine-Tuning. We assess QUAD's fine-tuning capabilities by adapting quantized models on downstream tasks. Zero-shot results for Llama-3 and Qwen-2.5 models fine-tuned on the Alpaca dataset (Tables 1, 2, and 5) show that parameter-efficient fine-tuning with QUAD achieves 98.4% to 100.0% of the original model's accuracy. Meanwhile, comparisons with QLoRA (Dettmers et al., 2023) (Table 4) on Qwen-2.5 models fine-tuned for Meta-Math-QA and Code-Feedback tasks reveal that QUAD outperforms QLoRA on GSM8K and HumanEval datasets.

Efficiency Analysis. We assessed the prefill speed of Llama-3-8B quantized with QuaRot and QUAD, setting the sequence length to 2048. The experimental results, illustrated in Figure 5, reveal approximately 1.95× and 2.05× speedups compared to the FP16 baseline when using QUAD and QuaRot, respectively. This indicates that the additional parameters introduced by QUAD incur only about 5% inference overhead. In our experiment, we additionally found that for models where the

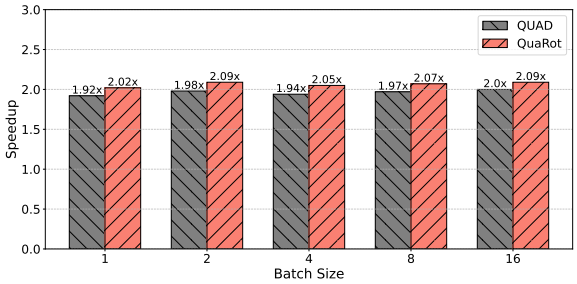

Figure 5: Speedup of QUAD/QuaRot on Llama-3-8B under sequence length 2048.

number of attention heads is not a power of 2, the online Hadamard transform leads to high latency, so for such models, we explore how to eliminate the online Hadamard transform in Appendix E.

6 Conclusion

In this work, we address the challenge of quantizing large language models (LLMs) by proposing QUAD (Quantization with Activation Decomposition). This framework effectively suppresses activation outliers through singular value decomposition (SVD). By decomposing activations into outlier-free components and retaining critical outlier dimensions in full precision, QUAD enables 4-bit quantization of weights and activations while maintaining high accuracy. Our method is compatible with existing rotation-based quantization techniques and introduces a parameter-efficient fine-tuning strategy to narrow the gap between quantized and full-precision models. Experiments on different LLMs demonstrate that QUAD preserves 95–97% accuracy of the full-precision baseline with W4A4 quantization and fine-tuning. The key contributions of QUAD include (1) a novel approach to handle activation outliers and seamless integration with rotation methods and (2) a parameter-efficient fine-tuning mechanism to enhance quantization performance.

7 Limitation

Although QUAD enhances model performance through mixed-precision quantization, the incorporation of mixed-precision matrix computations and the inability to generate an optimal fusion kernel using CUTLASS lead to a reduction in inference speed. This highlights the need for developing a more efficient, mixed-precision matrix multiplication-oriented kernel in future work. In addition, QUAD increases the dimensionality of the model's hidden size, which is not fully compatible with the model structure definition used in the existing framework, like Huggingface Transformers. As a result, additional modifications are required to integrate QUAD into current frameworks.

References

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024a. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. 2023. Quik: Towards end-to-end 4-bit inference on generative large language models. *arXiv preprint arXiv:2310.09259*.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024b. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240.

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, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Yuji Chai, John Gkountouras, Glenn G. Ko, David Brooks, and Gu-Yeon Wei. 2023. Int2.1: Towards fine-tunable quantized large language models with error correction through low-rank adaptation. *Preprint*, arXiv:2306.08162.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. 2023. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, et al. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2024. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*.

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. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. 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. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. 2024. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *arXiv preprint arXiv:2402.10631*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Han Guo, Philip Greengard, Eric P. Xing, and Yoon Kim. 2024. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning. *Preprint*, arXiv:2311.12023.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Alston S Householder. 1958. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4):339–342.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. 2025. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. *Preprint*, arXiv:2501.13987.

Yuxuan Hu, Jing Zhang, Zhe Zhao, Chen Zhao, Xiaodong Chen, Cuiping Li, and Hong Chen. 2024. sp^3: Enhancing structured pruning via PCA projection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3150–3170, Bangkok, Thailand. Association for Computational Linguistics.

Xijie Huang, Zechun Liu, Shih-Yang Liu, and Kwang-Ting Cheng. 2024. Rolora: Fine-tuning rotated outlier-free llms for effective weight-activation quantization. *Preprint*, arXiv:2407.08044.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Preprint*, arXiv:2305.14152.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024a. Codes: Towards building open-source language models for text-to-sql. *Proc. ACM Manag. Data*, 2(3).

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. 2024b. Svdqunat: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*.

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.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2024a. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800.

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

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024c. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. 2024b. Vptq: Extreme low-bit vector post-training quantization for large language models. *arXiv preprint arXiv:2409.17066*.

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. *arXiv preprint arXiv:2305.17888*.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024c. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. 2024. Affinequant: Affine transformation quantization for large language models. *arXiv preprint arXiv:2403.12544*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2025. Pissa: Principal singular values and singular vectors adaptation of large language models. *Preprint*, arXiv:2404.02948.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *Preprint*, arXiv:1606.06031.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Utkarsh Saxena, Sayeh Sharify, Kaushik Roy, and Xin Wang. 2025. Resq: Mixed-precision quantization of large language models with low-rank residuals. *Preprint*, arXiv:2412.14363.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. 2025. Flatquant: Flatness matters for llm quantization. *Preprint*, arXiv:2410.09426.

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.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. 2023. *CUTLASS*.

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

Mart Van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. 2024. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Hongyu Wang, Shuming Ma, and Furu Wei. 2024a. Bitnet a4. 8: 4-bit activations for 1-bit llms. *arXiv preprint arXiv:2411.04965*.

Lei Wang, Yu Cheng, Yining Shi, Zhengju Tang, Zhiwen Mo, Wenhao Xie, Lingxiao Ma, Yuqing Xia, Jilong Xue, Fan Yang, and Zhi Yang. 2025a. Tilelang: A composable tiled programming model for ai systems. *Preprint*, arXiv:2504.17577.

Shaowen Wang, Linxi Yu, and Jian Li. 2024b. Lora-ga: Low-rank adaptation with gradient approximation. *Preprint*, arXiv:2407.05000.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. 2025b. Lora-pro: Are low-rank adapters properly optimized? *Preprint*, arXiv:2407.18242.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

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.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2025. Opencodeinterpreter: Integrating code generation with execution and refinement. *Preprint*, arXiv:2402.14658.

Kan Zhu, Yilong Zhao, Liangyu Zhao, Gefei Zuo, Yile Gu, Dedong Xie, Yufei Gao, Qinyu Xu, Tian Tang, Zihao Ye, et al. 2024. Nanoflow: Towards optimal large language model serving throughput. *arXiv preprint arXiv:2408.12757*.

A QUAD on the Attention Module

Figure 6 shows the result of applying QUAD on the attention module. Based on the original Attention module, we fuse the projection matrix P, the rotation matrix Q, and the scaling factor (α) on the left side of the matrices W_q, W_k, and W_v, and additionally fuse the Hadamard matrix H_{head} on the right side of W_v.

B Hadamard Transform

A Hadamard matrix is an orthogonal matrix whose entries belong to $\{+1, -1\}$. A Walsh-Hadamard matrix is a square matrix of size $d = 2^n$, defined recursively as:

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \text{and} \quad H_{2^n} = H_2 \otimes H_{2^{n-1}},$$

where $\otimes$ represents the Kronecker product. These definitions underpin the Walsh-Hadamard transform, which efficiently computes the matrix-vector product Hx in $O(d \log_2(d))$ operations.

Furthermore, prior work in QuaRot (Ashkboos et al., 2024b) and QUIP (Chee et al., 2023) demonstrates that applying the Hadamard transform to tensors reduces the incoherence of weight matrices and activations. Specifically, a weight matrix W is considered μ-incoherent if $\max(W) \leq \mu \cdot \|W\|_F \cdot \frac{1}{\sqrt{mn}}$, where $\max(W)$ denotes the element-wise maximum of the matrix, and mn represents the total number of elements in W. A weight matrix with high incoherence is difficult to quantize because its largest elements stand out significantly in magnitude compared to the average entries, acting as outliers. Experiment results from QuaRot (Ashkboos et al., 2024b) and QUIP (Chee et al., 2023) demonstrated that multiplying such a matrix on both the left and right by orthogonal matrices can reduce its incoherence, thereby making it more amenable to quantization. In this work, we follow the Hadamard transform used in these works (Ashkboos et al., 2024b; Chee et al., 2023; Liu et al., 2024c; Hu et al., 2025) to smooth the weights and activations, thus reducing the quantization difficulty.

C Proofs

C.1 Proof about equivalence transformation

The RMSNorm module can be formalized as $\text{RMSNorm}(x) = \frac{x}{\|x\|}(\alpha)$. In this work, we

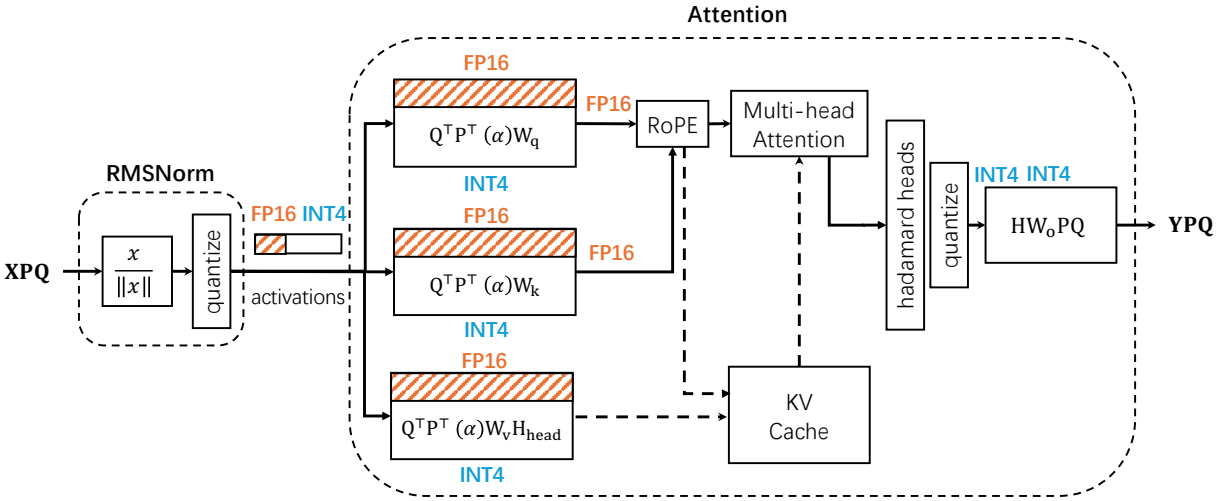

Figure 6: The Attention module of the transformer layer after applying QUAD transformation.

first fuse the scaling factor (α) with the adjacent weight matrix W, such that $\text{RMSNorm}(x)W = \left(\frac{x}{\|x\|}\right)((\alpha)W) = \hat{\text{RMSNorm}}(x)\hat{W}$.

Proposition. Thus, suppose $\text{RMSNorm}(x) = \frac{x}{\|x\|}$, then we have

$$\text{RMSNorm}(x) = \text{RMSNorm}(xQ^\top)Q.$$

Proof.

$$\begin{aligned}
\text{RMSNorm}(xQ^\top)Q &= \frac{xQ^\top}{\|xQ^\top\|}Q \\
&= \frac{xQ^\top Q}{\sqrt{xQ^\top Qx^\top}} \\
&= \frac{x}{\sqrt{xx^\top}} \\
&= \frac{x}{\|x\|} \\
&= \text{RMSNorm}(x)
\end{aligned}$$

□

C.2 Proof about projection matrix

Proposition.

$$PP^\top = I.$$

Proof.

$$\begin{aligned}
PP^\top &= \left(U_{1:r}, I - \sum_{i=1}^{r} U_i U_i^\top\right)\left(U_{1:r}, I - \sum_{i=1}^{r} U_i U_i^\top\right)^\top \\
&= \left(U_{1:r}, I - \sum_{i=1}^{r} U_i U_i^\top\right)\begin{pmatrix} U_{1:r}^\top, \\ I - \sum_{i=1}^{r} U_i U_i^\top \end{pmatrix} \\
&= \left(U_{1:r}U_{1:r}^\top + (I - \sum_{i=1}^{r} U_i U_i^\top)(I - \sum_{i=1}^{r} U_i U_i^\top)\right) \\
&= \left(\sum_{i=1}^{r} U_i U_i^\top + I - 2\sum_{i=1}^{r} U_i U_i^\top + \sum_{i=1}^{r} U_i U_i^\top\right) \\
&= I
\end{aligned}$$

□

C.3 Proof of Proposition 4.1

Proposition. *The quantization error can be decomposed as follows*:

$$E(X, W) \leq \|X\|_F \|W - Q(W)\|_F + \|X - Q(X)\|_F \|Q(W)\|_F.$$

Proof.

$$\begin{aligned}
&E(X, W) \\
&= \|XW - Q(X)Q(W)\|_F \\
&= \|XW - XQ(W) + XQ(W) - Q(X)Q(W)\| \\
&\leq \|X(W - Q(W))\|_F + \|(X - Q(X))Q(W)\|_F \\
&\leq \|X\|_F \|W - Q(W)\|_F + \|X - Q(X)\|_F \|Q(W)\|_F.
\end{aligned}$$

□

C.4 Proof of Proposition 4.2

Proposition. *For the round-to-nearest (RTN) quantization method Q, if the tensor X follows a normal distribution, we have*:

$$\mathbb{E}\left[\|X - Q(X)\|_F\right] \leq \frac{\sqrt{\log(\text{size}(X)\pi)}}{q_{\max}} \mathbb{E}\left[\|X\|_F\right],$$

The proof of proposition 4.2 can be found in Section A.2 of SVDQuant (Li et al., 2024b). For ease of readability, we also give the full proof here.

Proof.

$$\begin{aligned}\|X - Q(X)\|_F &= \|X - s_X \cdot Q_X\|_F \\ &= \left\| s_X \cdot \frac{X}{s_X} - s_X \cdot \text{round}\left(\frac{X}{s_X}\right)\right\|_F \\ &= |s_X| \left\| \frac{X}{s_X} - \text{round}(\frac{X}{s_X})\right\|_F.\end{aligned}$$

So,

$$\begin{aligned}\mathbb{E}\left[\|X - Q(X)\|_F\right] &\leq \mathbb{E}\left[|s_X|\right] \sqrt{\text{size}(X)} \\ &= \frac{\sqrt{\text{size}(X)}}{q_{\max}} \cdot \mathbb{E}\left[\max(|X|)\right] \\ &\leq \frac{\sqrt{\text{size}(X)}}{q_{\max}} \cdot \sigma\sqrt{2\log(\text{size}(X))}\end{aligned} \tag{5}$$

where σ is the standard deviation of the normal distribution. Equation (5) comes from the maximal inequality of Gaussian variables (Lemma 2.3 in Massart (2007)).

Meanwhile,

$$\begin{aligned}\mathbb{E}\left[\|X\|_F\right] &= \mathbb{E}\left[\sqrt{\sum_{x\in X} x^2}\right] \\ &\geq \mathbb{E}\left[\frac{\sum_{x\in X}|x|}{\sqrt{\text{size}(X)}}\right] \qquad (6) \\ &= \sigma\sqrt{\frac{2\text{size}(X)}{\pi}}, \qquad (7)\end{aligned}$$

where Equation (6) comes from the Cauchy-Schwarz inequality, and Equation (7) comes from the expectation of the half-normal distribution.

Thus, we have

$$\begin{aligned}\mathbb{E}\left[\|X - Q(X)\|_F\right] &\leq \frac{\sqrt{\text{size}(X)}}{q_{\max}} \cdot \sigma\sqrt{2\log(\text{size}(X))} \\ &\leq \frac{\sqrt{\log(\text{size}(X)\pi)}}{q_{\max}} \mathbb{E}\left[\|X\|_F\right].\end{aligned}$$

□

C.5 Proof of Proposition 4.3

Proposition. *Suppose the gradients of the weights W and W_r are ∇W and ∇W_r, then we have*:

$$X\nabla W = XX^\top \nabla Y \approx X_r \nabla W_r.$$

Proof. Consider the singular value decomposition (SVD) of the matrix $X^\top$, which is given as $X^\top = U\Sigma V^\top$. From this, we can derive the SVDs of $XX^\top$ and $X^\top X$:

$$\text{SVD}(XX^\top) = V\Sigma^2 V^\top, \text{SVD}(X^\top X) = U\Sigma^2 U^\top.$$

According to the Eckart-Young-Mirsky theorem, a matrix can be approximated optimally using its singular value decomposition. Specifically, for any matrix D, the best rank-r approximation is given by $U_{1:r}\Sigma_{1:r}V_{1:r}^\top$, where $\text{SVD}(D) = U\Sigma V^\top$. Therefore, the optimal rank-r approximation of $XX^\top$ is $V_{1:r}\Sigma_{1:r}^2 V_{1:r}^\top$.

Let $X_r = XU_{1:r}$. Then, we compute $X_r X_r^\top$ as follows:

$$\begin{aligned}X_r X_r^\top &= XU_{1:r}U_{1:r}^\top X^\top \\ &= V\Sigma U^\top U_{1:r}U_{1:r}^\top U\Sigma V^\top \\ &= V\Sigma \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix} \Sigma V^\top \\ &= V_{1:r}\Sigma_{1:r}^2 V_{1:r}^\top.\end{aligned}$$

Thus, $X_r X_r^\top$ is the optimal rank-r approximation of $XX^\top$. □

D Additional Experiment Results

Table 7 and 10 present the experimental results for QUAD on the Llama-2 model. The results indicate that QUAD outperforms QuaRot while QUAD-F outperforms SpinQuant, OSTQuant, and FlatQuant, underscoring its effectiveness in achieving higher accuracy in zero-shot tasks. Furthermore, we compared QUAD with additional baselines on Llama-2-7B, Llama-2-13B, and Llama-3-8B, and the corresponding experimental results are detailed in Table 11. Existing studies (Wang et al., 2024a) have shown that intermediate representations within the model's feed-forward network module tend to contain more outliers, which makes them particularly challenging to quantize effectively. To address this, using higher precision for these intermediate representations offers a better trade-off between model accuracy and inference speed. Thus, we also present experimental

results of this mixed-precision strategy. The experiment results are shown in Table 7, 8 and 9. Here, "W4A4/A8" denotes 8-bit quantization for intermediate representations of the FFN module and 4-bit quantization for other weights and activations.

E Eliminating Online Hadamard Transform

For the Attention module, both QUAD and existing approaches smooth the scale-dot-product-attention (SDPA) output using the online Hadamard transform. This approach is efficient for models where the number of attention heads is a power of 2, such as Llama-2-7B and Llama-3-8B. However, for models that do not meet this condition, such as the Qwen-2.5-7B and Llama-2-13B, it introduces substantial latency. As illustrated in Figure 7, under the W4A4/A8 quantization schema, employing the online Hadamard transform results in slower inference speeds compared to the baseline using FP16 precision. To address this issue, inspired by SVDQuant, we experimented with replacing the online Hadamard transform with full-precision low-rank branches. Specifically, for the Hadamard transform, we have:

$$F_{had}(X, W) = Q(\text{Hadamard}(X))Q(HW),$$

where Q denotes quantization. In the model, we replaced F_{had} with F_{LoRA}, which corresponds to the following expression:

$$F_{LoRA}(X, W', L, R) = Q(Xs^{-1})Q(sW') + Xs^{-1}LR,$$

where $s_i = \max(|X_i|)^{0.25}$, $L = U$, $R = \Sigma V^\top$, $W' = W - LR$, and $U\Sigma V = \text{SVD}(sW)$. As shown in Figure 7, replacing the Hadamard transform with LoRA yields a significant improvement in the model's inference speed, achieving approximately $1.7\times$ speedup relative to the baseline. Additionally, we compared the performance of the model under the two approaches, and the experimental results are presented in Table 6. These results demonstrate that the use of low-rank branching can achieve competitive performance with the Hadamard transform within the W4A4/A8 quantization schema.

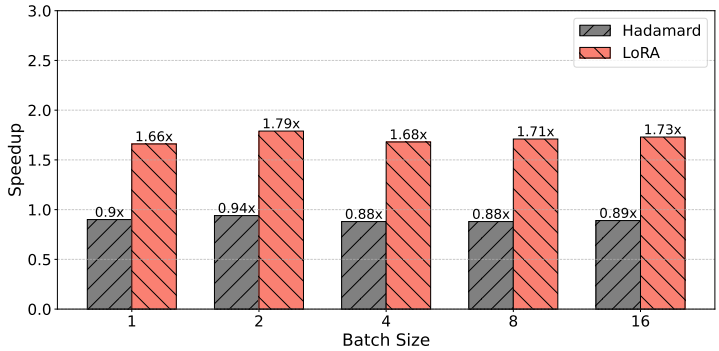

Figure 7: Speedup of QUAD-Hadamard/LoRA on Qwen-2.5-7B under sequence length 2048 and W4A4/A8 quantization schema.

Model	Method	PQ	WG	HS	A-e	A-c	LA	Avg.
Llama-2-7B	Hadamard	78.07	68.75	74.72	71.97	43.26	73.94	68.45
	LoRA	79.27	68.59	74.27	72.90	43.34	72.44	68.47
Llama-2-13B	Hadamard	79.71	72.30	78.11	75.63	47.87	76.56	71.70
	LoRA	79.27	71.43	77.81	76.56	48.12	77.78	71.83
Llama-3-8B	Hadamard	80.09	71.35	77.12	73.82	47.53	74.35	70.71
	LoRA	78.67	72.06	76.73	76.77	48.98	74.36	71.26
Qwen-2.5-1.5B	Hadamard	73.88	60.22	64.38	70.54	42.83	59.31	61.86
	LoRA	72.85	61.48	64.06	70.92	41.64	57.27	61.37
Qwen-2.5-3B	Hadamard	76.50	66.30	70.76	71.21	45.99	63.48	65.71
	LoRA	76.44	63.85	70.85	73.44	46.42	62.78	65.63
Qwen-2.5-7B	Hadamard	79.43	69.61	76.72	77.10	50.51	69.88	70.54
	LoRA	77.91	69.61	76.44	76.01	49.74	68.97	69.78

Table 6: Comparison of Hadamard transform and low-rank branch on LLAMA and Qwen models with W4A4/A8 quantization schema.

Model	Method	Precision	PQ	WG	HS	A-e	A-c	LA	Avg.
Llama-2-7B	Baseline	FP16	79.11	69.06	75.99	74.54	46.33	73.84	69.81
	QuaRot	W4A4	77.80	65.04	73.23	69.82	42.66	71.53	66.68
	QUAD	W4A4	77.58	68.43	73.73	71.68	41.64	73.34	67.73
	QUAD	W4A4/A8	78.07	68.75	74.72	71.97	43.26	73.94	68.45
Llama-2-13B	Baseline	FP16	80.52	72.38	79.38	77.44	49.15	76.77	72.61
	QuaRot	W4A4	77.69	69.22	76.02	74.54	47.78	74.48	69.96
	QUAD	W4A4	79.00	71.11	77.45	75.08	46.76	75.86	70.88
	QUAD	W4A4/A8	79.71	72.30	78.11	75.63	47.87	76.56	71.70

Table 7: Zero-shot accuracy of LLAMA-2 models.

Model	Method	Precision	PQ	WG	HS	A-e	A-c	LA	Avg.
Llama-3.2-1B	Baseline	FP16	74.59	60.38	63.68	60.48	36.35	62.93	59.74
	QUAD	W4A4	69.80	56.20	57.33	54.71	31.40	51.29	53.46
	QUAD	W4A4/A8	72.80	58.72	60.71	57.15	32.85	58.12	56.73
Llama-3.2-3B	Baseline	FP16	77.48	69.93	73.60	71.68	45.90	70.50	68.18
	QUAD	W4A4	74.65	65.75	70.01	65.11	42.41	65.48	63.90
	QUAD	W4A4/A8	75.24	65.51	71.60	67.68	43.94	68.62	65.43
Llama-3-8B	Baseline	FP16	80.79	73.16	79.16	77.74	53.24	75.74	73.30
	QUAD	W4A4	77.48	68.43	75.22	73.74	46.50	71.18	68.76
	QUAD	W4A4/A8	80.09	71.35	77.12	73.82	47.53	74.35	70.71

Table 8: Zero-shot accuracy of LLAMA-3 models.

Model	Method	Precision	PQ	WG	HS	A-e	A-c	LA	Avg.
Qwen2.5-1.5B	Baseline	FP16	75.95	63.46	67.73	71.68	45.48	62.06	64.39
	QUAD	W4A4	72.36	59.19	62.70	68.81	41.55	56.01	60.10
	QUAD	W4A4/A8	73.88	60.22	64.38	70.54	42.83	59.31	61.86
Qwen2.5-3B	Baseline	FP16	78.40	68.51	73.59	72.98	46.93	66.93	67.89
	QUAD	W4A4	75.57	62.67	69.14	70.12	44.03	60.45	63.66
	QUAD	W4A4/A8	76.50	66.30	70.76	71.21	45.99	63.48	65.71
Qwen2.5-7B	Baseline	FP16	79.98	72.69	78.90	77.31	51.11	71.71	71.95
	QUAD	W4A4	78.89	68.67	75.66	74.75	48.46	67.59	69.00
	QUAD	W4A4/A8	79.43	69.61	76.72	77.10	50.51	69.88	70.54

Table 9: Zero-shot accuracy of Qwen-2.5 models.

Model	Method	PQ	WG	HS	A-e	A-c	LA	Avg.
Llama-2-7B	Baseline	79.11	69.06	75.99	74.54	46.33	73.84	69.81
	SpinQuant	76.88	65.82	73.51	71.08	40.44	70.66	66.40
	OSTQuant	77.42	66.77	73.14	72.56	42.92	61.76	65.76
	FlatQuant	78.94	66.46	74.26	72.60	44.03	71.59	67.98
	FlatQUAD	78.29	67.01	74.46	73.70	43.34	72.93	68.30
Llama-2-13B	Baseline	80.63	71.90	79.39	77.53	49.15	76.62	72.54
	SpinQuant	78.29	67.72	75.22	74.49	46.67	72.19	69.10
	OSTQuant	75.14	68.03	76.71	75.21	47.10	75.14	69.55
	FlatQuant	79.38	71.19	78.22	76.35	47.44	76.29	71.48
	FlatQUAD	79.33	71.27	77.20	77.02	49.49	76.11	71.74

Table 10: Comparison of QUAD and advanced rotation-based methods on LLAMA models with W4A4 quantization schema.

Model	#Bits W-A-KV	Method	A-c	A-e	BQ	HS	LA	OQ	PQ	SQ	WG	Avg.
Llama-2-7B	16-16-16	Baseline	46.33	74.54	77.71	75.99	73.84	44.20	79.11	46.11	69.06	65.21
	4-16-16	RTN	42.15	67.59	73.06	72.34	67.18	41.80	76.50	44.11	66.69	61.27
		SmoothQuant	39.59	65.19	69.82	68.84	62.27	40.20	75.95	44.17	63.85	58.88
		GPTQ	42.49	69.53	61.31	73.83	67.61	42.40	77.64	44.52	68.43	60.86
		OmniQuant	42.49	71.00	74.34	73.85	70.70	44.20	78.40	44.93	68.82	63.19
		AWQ	44.11	70.75	78.07	74.98	70.68	43.80	78.13	45.14	69.38	63.89
		QuaRot	43.94	73.15	76.97	74.87	73.06	44.00	78.24	45.09	69.38	64.30
		QUAD	44.11	73.19	76.51	75.30	74.27	43.80	78.62	46.06	70.01	64.65
	4-4-16	RTN	25.34	28.03	50.52	27.71	1.01	26.20	50.82	33.93	48.38	32.44
		SmoothQuant	28.33	26.39	49.39	27.28	1.18	23.40	48.80	33.62	50.75	32.13
		GPTQ	24.40	28.70	51.62	28.66	1.36	24.60	51.14	34.49	49.49	32.72
		QuaRot	42.32	69.65	74.77	72.91	70.81	39.80	77.20	43.55	65.82	61.87
		QUAD	41.64	71.68	74.22	73.73	73.34	43.00	77.58	45.70	68.43	63.26
	4-4-4	RTN	27.22	27.06	50.83	27.34	0.93	25.80	49.51	34.85	50.51	32.67
		SmoothQuant	26.37	25.63	47.71	27.05	1.11	26.40	51.90	34.49	48.38	32.12
		GPTQ	26.96	27.65	52.84	28.83	1.63	29.20	49.62	35.11	49.80	33.52
		OmniQuant	31.40	53.75	63.79	55.06	35.63	34.40	66.59	40.28	54.70	48.40
		QuaRot	41.43	69.32	74.19	72.50	70.66	39.80	77.42	43.35	64.64	61.48
		QUAD	41.64	71.68	74.22	73.73	73.34	43.00	77.58	45.70	68.43	63.26
Llama-2-13B	16-16-16	Baseline	49.15	77.53	80.58	79.39	76.62	45.20	80.63	47.49	71.90	67.61
	4-16-16	RTN	42.92	66.54	71.38	66.62	68.99	39.40	76.93	44.06	65.35	60.24
		SmoothQuant	46.25	70.45	74.92	69.16	70.49	39.80	77.86	45.14	64.17	62.03
		GPTQ	49.63	73.95	74.83	73.77	73.20	42.40	78.51	45.50	70.64	64.71
		OmniQuant	48.29	75.42	77.92	77.80	75.59	45.20	80.41	46.62	70.17	66.38
		AWQ	48.63	78.16	78.81	78.48	75.20	45.00	79.54	46.21	72.45	66.94
		QuaRot	49.15	76.26	80.46	78.17	76.50	45.40	80.03	45.50	71.11	66.95
		QUAD	47.61	75.42	79.24	78.87	77.28	45.40	79.65	45.96	72.30	66.86
	4-4-16	RTN	27.99	26.81	38.50	26.08	0.00	23.60	48.20	34.90	51.62	30.86
		SmoothQuant	24.49	35.06	47.98	30.87	3.67	26.20	55.01	35.31	49.72	34.26
		GPTQ	27.82	26.77	37.92	25.67	0.00	21.80	47.77	35.11	48.15	30.11
		QuaRot	46.42	73.86	78.10	75.68	74.31	43.00	79.05	44.37	71.35	65.13
		QUAD	46.76	75.08	78.69	77.45	75.86	43.40	79.00	45.14	71.11	65.83
	4-4-4	RTN	27.82	26.52	38.38	26.27	0.02	26.00	49.78	34.39	49.17	30.93
		SmoothQuant	24.49	33.00	45.84	30.70	2.70	23.80	53.81	34.80	51.07	33.36
		GPTQ	27.90	26.39	37.95	26.16	0.00	27.00	48.26	34.39	50.43	30.94
		OmniQuant	32.85	55.13	64.34	60.13	42.85	33.40	68.17	39.76	56.51	50.35
		QuaRot	47.27	73.91	78.41	75.33	73.53	43.80	79.27	45.85	69.06	65.16
		QUAD	46.50	74.96	78.47	77.32	76.01	44.20	80.03	45.09	71.27	65.98
Llama-3-8B	16-16-16	Baseline	53.50	77.74	81.10	79.18	75.74	44.80	80.63	47.08	73.01	68.09
	4-16-16	RTN	48.98	73.23	72.75	75.90	63.85	43.20	78.40	43.81	73.16	63.70
		SmoothQuant	47.44	72.35	72.11	74.92	62.41	43.00	77.69	43.91	71.27	62.79
		GPTQ	49.74	72.52	71.28	68.34	46.69	43.60	78.78	46.47	71.82	61.03
		OmniQuant	50.09	74.54	79.51	76.92	70.31	43.80	79.54	44.52	71.74	65.66
		AWQ	52.22	76.68	80.31	77.51	74.81	44.20	79.60	46.26	71.67	67.03
		QuaRot	51.88	77.53	79.60	77.87	74.11	44.40	80.14	46.37	73.56	67.27
		QUAD	50.77	75.34	80.58	77.93	75.63	45.00	80.63	46.26	73.80	67.33
	4-4-16	RTN	23.72	30.89	46.30	31.26	3.03	27.60	52.72	35.26	50.04	33.42
		SmoothQuant	23.29	28.28	48.93	29.19	1.57	28.60	54.46	33.37	49.64	33.04
		GPTQ	23.46	32.07	43.79	30.10	2.41	28.00	53.97	34.14	48.86	32.98
		QuaRot	42.66	67.26	73.73	73.60	67.42	43.00	76.61	45.04	65.90	61.69
		QUAD	46.50	73.74	76.02	75.22	71.18	43.00	77.48	44.42	68.43	64.00
	4-4-4	RTN	23.72	30.56	46.18	29.83	2.70	28.60	52.45	34.39	50.20	33.18
		SmoothQuant	23.55	28.96	48.84	28.90	1.44	29.40	51.09	34.14	50.36	32.96
		GPTQ	23.38	32.74	44.34	29.72	2.39	29.80	54.95	34.75	51.30	33.71
		OmniQuant	22.87	30.35	41.53	31.11	1.86	25.40	53.37	34.08	50.43	32.33
		QuaRot	42.83	67.42	73.21	72.66	66.93	42.20	75.73	45.19	66.22	61.38
		QUAD	45.90	72.10	73.98	74.90	69.82	43.40	78.45	44.22	71.43	63.80

Table 11: Experiment results of LLAMA-2 and LLAMA-3 models.