KurTail : Kurtosis-based LLM Quantization

Anonymous ACL submission

Abstract

One challenge of quantizing a large language model (LLM) is the presence of outliers. Outliers often make uniform quantization schemes less effective, particularly in extreme cases such as 4-bit quantization. We introduce KurTail, a new post-training quantization (PTQ) scheme that leverages Kurtosis-based rotation to mitigate outliers in the activations of LLMs. Our method optimizes Kurtosis as a measure of tailedness. This approach enables the quantization of weights, activations, and the KV cache in 4 bits. We utilize layer-wise optimization, ensuring memory efficiency. KurTail outperforms existing quantization methods, offering a 13.3% boost in MMLU accuracy and a 15.5% boost in Wiki perplexity compared to QuaRot (Ashkboos et al., 2024b). It also outperforms SpinQuant (Liu et al., 2024) with a 2.6% MMLU gain and reduces perplexity by 2.9%, all while reducing the training cost. For comparison, learning the rotation using SpinQuant for Llama3-70B requires at least four NVIDIA H100 80GB GPUs, whereas our method requires only a single GPU, making it more accessible.

1 Introduction

Large language models (LLMs) have advanced significantly in recent years, showcasing remarkable performance and capabilities. As these models grow in size and complexity, the computational cost required for their deployment and inference has increased dramatically. Furthermore, with new inference time methods (OpenAI, 2024; Guo et al., 2025), enhancing inference speed (tokens per second) is increasingly important. This has shifted the focus toward accelerating model performance while reducing memory and computational requirements. An effective method to achieve this is post-training quantization (PTQ), which involves representing model weights and/or activations in lower numerical precisions. PTQ can significantly reduce the memory footprint and computational overhead and subsequently decrease latency and energy consumption, which are especially beneficial for inference on resource-constrained edge devices.

Serving a model involves two stages of *prefilling* and *generation*. During *prefilling*, the model processes the input prompt and stores the internal state, known as key-value (KV) caching. During *generation*, tokens are produced auto-regressively. Quantizing each stage offers distinct advantages for improving inference efficiency. KV-cache quantization reduces memory requirements and accelerates data movement, which enhances the *generation* stage, particularly in scenarios involving long-context inference. Weight quantization, on the other hand, reduces the memory footprint independently, and when it is combined with activation quantization, it also reduces the computational demands. However, activation quantization presents challenges due to large outliers in certain channels (Dettmers et al., 2022; Xiao et al., 2023), which limits the effectiveness of uniform integer quantization as it destroys the dynamic range of the activations. While channel-wise quantization can effectively address this issue, the lack of hardware support makes it computationally expensive in practice. Several methods have been proposed to address this challenge. Dettmers et al. (2022) and Ashkboos et al. (2023) advocate for mixed-precision computation in which they store some of the channels in higher precision and less sensitive channels in lower precision to balance accuracy and efficiency. Xiao et al. (2023) introduces channel-wise scaling into the layer normalization and the weights of linear layers. Ashkboos et al. (2024b) proposed random rotation which takes the advantage of the computational invariance framework (Ashkboos et al., 2024a) to mitigate the outliers problem.

We introduce *KurTail* – a novel approach to mitigating activation outliers by applying learnable

rotations[1] to the activations similar to SpinQuant (Liu et al., 2024). KurTail focuses on reducing the tail density of activations, captured by the Kurtosis. Unlike SpinQuant which requires expensive end-to-end training of the model's loss, we prove that layer-wise optimization of our Kurtosis loss is equivalent to end-to-end training. We perform layer-wise inference to cache activations, and then optimize the rotations based on the cache independently. As a result, KurTail can be implemented in a significantly more memory-efficient manner. For instance, while SpinQuant requires at least four NVIDIA H100 80GB GPUs to compute rotations for Llama3-70B, KurTail achieves the same with just a single GPU. Despite its lower computational requirements, KurTail outperforms existing methods in terms of perplexity and zero-shot reasoning tasks. KurTail outperforms existing quantization methods with a 13.3% increase in MMLU accuracy and a 15.5% decrease in Wiki perplexity compared to QuaRot(Ashkboos et al., 2024b). It also performs better than SpinQuant(Liu et al., 2024), achieving a 2.6% increase in MMLU accuracy and a 2.9% decrease in perplexity, all while reducing the cost of training the rotation. We also theoretically shed light on why rotations are preferable to arbitrary linear transformations.

2 Background

Post Training Quantization. Previous work on post-training quantization fits into two main groups: weight-only quantization (Frantar et al., 2022; Lin et al., 2024; Egiazarian et al., 2024; Tseng et al., 2024) and weight-activation quantization (Xiao et al., 2023; Dettmers et al., 2022; Ashkboos et al., 2024b; Liu et al., 2024). In weight only quantization, the weight are projected into a lower precision, such as 4 bits, 3 bits, or even less, and then de-quantized to higher precision before the actual computation, with all calculations still being done in high precision. Several studies (Xiao et al., 2023; Ashkboos et al., 2024b; Liu et al., 2024) attempted to introduce quantization methods for both weight and activation. They showed that uniform quantizing is impractical for large language models since they suffer from large outliers. To address this issue, Dettmers et al. (2022) proposed a mixed-precision approach for handling outliers at higher precision. Others (Xiao et al., 2023; Lin et al., 2024) proposed trading outliers between weights and activations by introducing a re-scaling paradigm. Tseng et al. (2024) introduced an incoherence processing method using random rotation matrices and applying vector quantization on the weights for compression, adding overhead to inference. QuaRot (Ashkboos et al., 2024b) was inspired by Tseng et al. (2024) and took advantage of the invariance framework proposed by Ashkboos et al. (2024a) introducing a rotation-based approach to compress and remove outliers from the activation space using a random Hadamard rotation. Later, SpinQuant (Liu et al., 2024) improves the results of QuaRot (Ashkboos et al., 2024b) by optimizing some of these rotations to minimize the cross-entropy loss through end-to-end training. While SpinQuant improves the results compared to QuaRot it suffers from a high computational cost for learning the rotations. We address this issue by introducing a novel loss for learning the rotations.

Uniform Quantization for k-bit Precision. For a given vector $\boldsymbol{x}$, uniform integer quantization reduces its continuous range of values to a finite set of discrete levels, enabling representation in lower precision. In k-bit quantization, the value range $[\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}]$ is divided into 2^k equal intervals. Each element $\boldsymbol{x}_i$ in $\boldsymbol{x}$ is mapped to its closest quantization level by $Q(x_i) = \text{round}\left(\frac{\boldsymbol{x}_i - b}{s}\right) \cdot s + b$. Here s is the scale factor or step size and b is the shift. The values of s and b depend on the specific quantization scheme. In symmetric quantization, the range is assumed to be symmetric around zero. Therefore, $b = 0$, and $s = \frac{\max(|\boldsymbol{x}_{\max}|, |\boldsymbol{x}_{\min}|)}{2^{k-1}-1}$. In asymmetric quantization, the range is not assumed to be centered at zero and therefore, $b = \min(\boldsymbol{x}), \quad s = \frac{\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}}{2^k - 1}$. Given $\boldsymbol{x}$ sampled from some distribution the expected mean-squared error (MSE) between the quantized and the original values is:

$$\text{MSE}(\boldsymbol{x}, Q) = \mathbb{E}\left[(\boldsymbol{x} - Q(\boldsymbol{x}))^2\right] \tag{1}$$

Definition 2.1. Quantization Sensitivity (Chmiel et al., 2020) For a given distribution and sample vector $\boldsymbol{x}$, let $\tilde{s}$ denote the optimal quantization step size where $\tilde{s}$ minimizes the quantization error, and let $Q_{\tilde{s}}(\boldsymbol{x})$ represent the optimal quantizer. Quantization sensitivity $\Gamma(\boldsymbol{x}, \epsilon)$ is defined as the increase in the mean squared error (MSE) caused by a small perturbation $\epsilon > 0$ in the quantization step size s around $\tilde{s}$, such that $|s - \tilde{s}| = \epsilon$. Specifically, the sensitivity is given by:

$$\Gamma(\boldsymbol{x}, \epsilon) = |\text{MSE}(\boldsymbol{x}, s) - \text{MSE}(\boldsymbol{x}, \tilde{s})| \tag{2}$$

[1]We use rotation to refer to any orthogonal transformation.

Theorem 2.2. *(Chmiel et al., 2020) Considering $\boldsymbol{x}_U$ and $\boldsymbol{x}_N$ be continuous random variables with uniform and normal distributions. Then, for any given $\varepsilon > 0$, the quantization sensitivity $\Gamma(\boldsymbol{x}, \varepsilon)$ satisfies $\Gamma(\boldsymbol{x}_U, \varepsilon) < \Gamma(\boldsymbol{x}_N, \varepsilon)$.*

This theorem indicates that, compared to the typical normal distribution, the uniform distribution is more robust to changes in the quantization step size s. Therefore, it becomes apparent that there is great benefit in adjusting the distribution of the activations and weight to get closer to uniform distribution. This implies that the uniform distribution is a perfect fit for uniform quantization. It can also be shown that the optimal scaling $\tilde{s}$ for the uniform distribution is equal to $\tilde{s} = \frac{x_{\max} - x_{\min}}{2^k - 1}$. Chmiel et al. (2020) also show that the optimal step size for a uniform distribution closely approximates the most robust quantization (least sensitive step size).

Kurtosis. Kurtosis is a statistical measure that describes the degree of tailedness in the distribution of a dataset. It helps determine whether the data have heavy or light tails compared to a normal distribution. Mathematically, Kurtosis is defined as the standardized fourth moment of a population around its mean, and it is calculated using

$$\kappa = \frac{\mathbb{E}[(x-\mu)^4]}{(\mathbb{E}[(x-\mu)^2])^2} = \frac{\mu_4}{\sigma^4} \tag{3}$$

where μ is the mean, μ_4 is the fourth moment about the mean, and σ is the standard deviation. The Kurtosis of a normal distribution is 3. $\kappa > 3$ is characterized by heavy tails and a sharp peak, indicating greater tail density than a normal distribution (e.g. the Laplacian distribution). We have a shift of mass from the shoulders to both the tails and the center. On the contrary, $\kappa < 3$ is a sign of light tails and a flatter distribution (e.g. uniform or beta distribution) caused by mass moving from the tails and center to the shoulders. Banner et al. (2019) demonstrate that deep neural network weights and activations typically follow Gaussian or Laplace distributions. Furthermore, Dettmers et al. (2022) identifies the presence of extreme outliers in LLM parameters, which are critical for maintaining performance. Our key insight is that distributions with outliers exhibit high kurtosis, which measures the presence of extreme values. Therefore, by optimising the rotation to minimize the kurtosis we can bring the distribution closer to uniform. Uniform distribution is the desired distribution of the activations and weights for uniform quantization (§ 2), so we aim to move the distribution closer to uniform. Kurtosis serves two purposes: to encourage the distribution to resemble a uniform distribution, and to reduce the outliers. Our loss function is:

$$\mathcal{L}_\kappa = \frac{1}{L}\sum_{i=1}^{L} |\kappa(\bigoplus_{j=1}^{N} \boldsymbol{a}_{ij}) - \kappa_u| \tag{4}$$

where $\bigoplus$ denotes the concatenation of the activation of all tokens at that layer and κ_u is the Kurtosis of the uniform distribution.

2.1 Optimality of orthogonal transformations

There are two main reasons for using orthogonal transformations. First, when fusing the initial rotation $\boldsymbol{R}_1$, an orthogonal transformation is required to maintain invariance with respect to RMSNorm (see § 3), as shown by (Ashkboos et al., 2024b). In principle some of the transformation (i.e $\boldsymbol{R}_2$) can be any full rank matrix. We show that the quantization error is upper bounded by its condition number which is minimized for orthogonal transformations.

Lemma 2.3. *The k-bit quantization error of $\boldsymbol{X} \in \mathbb{R}^{N\times M}$ after a full rank transformation $\boldsymbol{T}$ is*

$$\left\|\boldsymbol{X} - Q(\boldsymbol{X}\boldsymbol{T})\boldsymbol{T}^{-1}\right\|_F \leq \frac{\|\boldsymbol{X}\|_F}{2^{k-1}-1}\sqrt{NM}\cdot \mathrm{cn}(\boldsymbol{T})$$

where $\mathrm{cn}(\boldsymbol{T})$ is the condition number of $\boldsymbol{T}$.

Corollary 2.4. *The upper bound on the quantization error is minimized when $\mathrm{cn}(\boldsymbol{T}) = 1$ and $\boldsymbol{T}$ is (a scalar multiple of) an orthogonal matrix.*

Intuitively, the quantization error of the transformed activation is inversely related to the smallest singular value of $\boldsymbol{T}$. To avoid amplifying the quantization error, it must not be smaller than one. An orthogonal transformation, where all singular values are equal to one, is well behaved.

2.2 End-to-End training

KurTail can be run layer-wise instead of end-to-end resulting in a computational benefit. We prove that end-to-end training and our layer-wise optimization converge to the same solution for certain families of models which include our current setting.

Proposition 2.5. *Let $\boldsymbol{H}_1, \boldsymbol{O}_1 = f(\boldsymbol{X}, \boldsymbol{W}_1, \boldsymbol{R}_1)$, and $\boldsymbol{H}_2, \boldsymbol{O}_2 = g(\boldsymbol{O}_1; \boldsymbol{W}_2; \boldsymbol{R}_2)$ where f, g are parameterized by $\boldsymbol{W}_1, \boldsymbol{R}_1$ and $\boldsymbol{W}_2, \boldsymbol{R}_2$. Given functional invariance of f and g, i.e. $\boldsymbol{O}_1 = \boldsymbol{O}_1'$ for any $\boldsymbol{O}_1', \boldsymbol{H}_1' = f(\boldsymbol{X}, \boldsymbol{W}_1, \boldsymbol{R}_1')$ and any orthogonal $\boldsymbol{R}_1'$ (and similarly for g), and given that the total loss is $\mathcal{L}(\boldsymbol{R}_1, \boldsymbol{R}_2) = \mathcal{L}_1(\boldsymbol{H}_1) + \mathcal{L}_2(\boldsymbol{H}_2)$, the independent*

minimization of each loss results in the same optimum as end-to-end: $\arg\min_{\boldsymbol{R}_1,\boldsymbol{R}_2} \mathcal{L}(\boldsymbol{R}_1, \boldsymbol{R}_2) = (\arg\min_{\boldsymbol{R}_1} \mathcal{L}_1(\boldsymbol{R}_1),\ \arg\min_{\boldsymbol{R}_2} \mathcal{L}_2(f; \boldsymbol{R}_2))$*, even though* $\boldsymbol{H}_2$ *implicitly depends on* $\boldsymbol{O}_1$.

Proposition 2.5 indicates that optimizing $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ end-to-end is equivalent to optimizing each separately since our loss and the model architecture satisfy the assumptions. Inductively, this holds for all layers. However, for the output of the MHSA and the FFN blocks we jointly optimize $\boldsymbol{R}_1$ using the activations from all layers by summing them since $\boldsymbol{R}_1$ shared across layers/losses (Fig. 3).

Quantization Sensitivity. We evaluate our method by measuring activation sensitivity both before and after applying rotations optimized with Kurtosis. We expect that after applying these rotations, the activation distribution will be closer to uniform, resulting in better quantization robustness. We empirically measure the sensitivity of the activation distribution before and after applying the rotation. We utilize the Llama3.1 8-B model and apply two rotation techniques: one using a random Hadamard transformation and another using a Kurtosis-optimized rotation. First, we compute the optimal scaling (Chmiel et al., 2020) for activation quantization and then calculate the quantization sensitivity based on Definition 2.1.

In Fig. 1, α indicates the fraction of the optimal step size used to analyze quantization sensitivity. The results show that the random Hadamard transformation reduces quantization sensitivity. Our Kurtosis-based method exhibits a bigger reduction in sensitivity, suggesting that it more effectively aligns the distribution with uniformity. Interestingly, we also observed that the sensitivity drop is strongest in the first layer compared to other layers for both methods. In Fig. 1 we compare layer 1 to layer 15, but this trend holds for deeper layers.

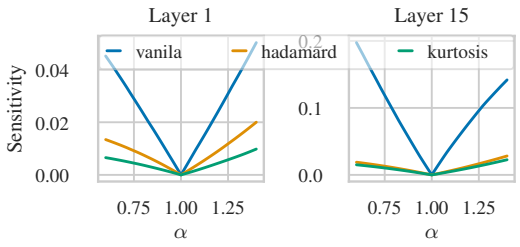

Figure 1: Empirical sensitivity of the MHSA input distribution across different rotations. α indicates the fraction of the optimal step size, i.e. sensitivity with step $\alpha \cdot s$.

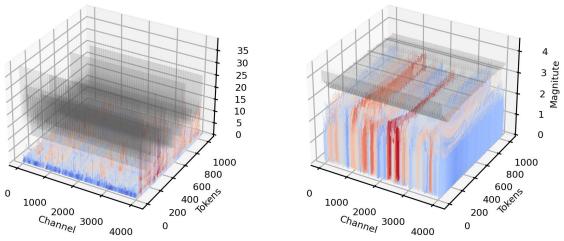

Figure 2: The input distribution of the MHSA blocks in the LLaMA3-8B model is shown before and after applying KurTail . Before rotation, some channels have noticeable outliers, which can disrupt the data balance. The rotated distribution allows for more accurate token-wise qunatization.

3 KurTail

Placement of the Rotations. Following the computational invariance theorem — as introduced by Elhage et al. (2023); Ashkboos et al. (2024a) and later utilized by QuaRot and SpinQuant — we adopted a similar framework to transform the activation functions at each layer. The placement of rotations is illustrated in Fig. 3. This figure depicts a single layer of a transformer model, where each square represents a computation block. The rotations are categorized into fusible rotations ($\boldsymbol{R}_1$ and $\boldsymbol{R}_2$) and online rotations ($\boldsymbol{R}_3$, $\boldsymbol{R}_4$, and $\boldsymbol{R}_5$). Fusible rotations do not add additional computational costs during inference since they can be merged with the model's original parameters. Specifically, we apply $\boldsymbol{R}_1$ to the left side of the token embedding, $\boldsymbol{W}_o$, and $\boldsymbol{W}_d$ within the MHSA and FFN blocks, respectively. The inverse of $\boldsymbol{R}_1$ is applied to the right side of $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v$ in the attention block, and $\boldsymbol{W}_{up}$, $\boldsymbol{W}_{gate}$ in the FFN block. Due to the residual connection, the exact same rotation must also be applied across subsequent layers (e.g., $\boldsymbol{X}\boldsymbol{R}_1 + \boldsymbol{Y}\boldsymbol{R}_1$ in one layer and $\boldsymbol{Y}\boldsymbol{R}_1 + \boldsymbol{X}_2\boldsymbol{R}_1$ in the next). The second fusible rotation, $\boldsymbol{R}_2$, is applied to the right side of $\boldsymbol{W}_v$, with its inverse applied to the left side of $\boldsymbol{W}_o$. This transformation improves the distribution of KV-caches and can vary across layers. The second group of rotations, $\boldsymbol{R}_3$, $\boldsymbol{R}_4$, and $\boldsymbol{R}_5$, are online which minimally increase the computational costs compared to the original model but they improve the performance. To mitigate this, we utilize random Hadamard matrices, which are computationally efficient, resulting in minimal overhead. For $\boldsymbol{R}_3$, the transformation is applied after each rotational positional encoding

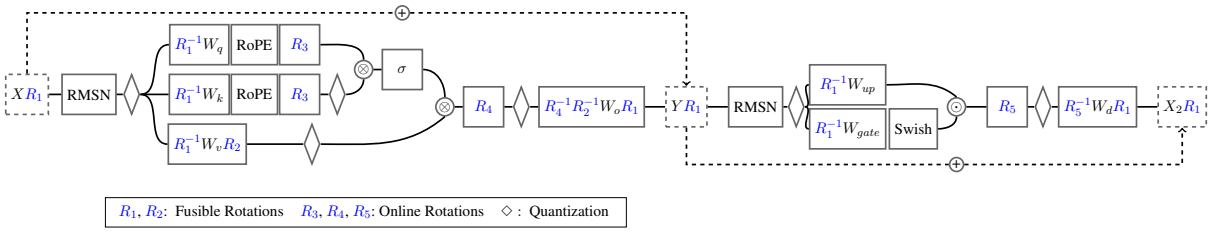

Figure 3: Diagram of a single-layer decoder network after applying rotations. Each block represents a computation unit. Blocks containing both blue and black indicate that the rotation is fused into the network without adding extra computation. In contrast, blocks with only the rotation signify additional computations during inference.

for queries and keys. Since the transpose of any orthogonal matrix equals its inverse, there is no need to add the inverse matrix explicitly. During the computation of attention scores, the term $\boldsymbol{Q}^T\boldsymbol{K}$ simplifies to $\boldsymbol{Q}^T\boldsymbol{R}_3^T\boldsymbol{R}_3\boldsymbol{K}$, effectively nullifying the impact of the rotation. For $\boldsymbol{R}_4$, we introduce the transformation after applying the softmax scores to the values and add the inverse in the subsequent linear layer. Similarly, $\boldsymbol{R}_5$ is implemented in the FFN block using the same approach.

Learning the Rotations. To discover the optimal rotations, we first run the vanilla model and store the inputs from both the MHSA and FFN blocks. Next, we create a small network consisting of a linear layer and an RMSNorm, designed to simulate the inputs of the MHSA and FFN blocks before quantization (Fig. 3). For optimization, we shuffle the stored input data from all transformer layers and both blocks and then train the rotation using Kurtosis loss. Since the optimization requires the rotations to remain within the orthogonal space, we use the Caley Adam (Li et al., 2020) optimizer to enforce this constraint. We train this small network for 100 iterations using 500 samples from the WikiText (Merity et al., 2016a) training set. In Table 7, we also did an ablation study on the different calibration size and datasets. After training, the resulting rotation is fused into the original network. For the $\boldsymbol{R}_2$, we apply a similar approach, but we removed the RMSNorm and just optimize the linear layer with the Kurtosis loss.

Optimization in the Orthogonal Space. As discussed in § 3, the transformation needs to be optimized in the orthogonal space to be consistent with a computational invariance framework. Therefore, we optimize all of the transformation matrices within the Stiefel Manifold (Li et al., 2020) i.e., the space of orthonormal matrices, using Caley Stochastic Gradient Descent (SGD) or Caley Adam (Li et al., 2020). For more detailed see (Li et al., 2020).

Training Cost. While quantization make the inference of large models feasible on consumer GPUs, finding the optimal rotation still requires substantial computational power. We address this by avoiding end-to-end fine-tuning. Since each multi-head attention and FFN is affected by $\boldsymbol{R}_1$, end-to-end approaches like SpinQuant cannot optimize the rotation layer by layer, and directly optimizing $\boldsymbol{R}_1$ via gradient descent requires loading the entire model, which is memory-intensive. Although SpinQuant reduces training costs by eliminating the need to store weight gradients and states, it still requires loading the full model into GPU memory. Our approach uses layer-wise inference, which eliminates the need to load all the network weight on the GPU at once. Then we store the activations for each layer. The we optimize the rotation with a Kurtosis loss. This significantly lowers GPU requirements—at most, a single NVIDIA H100 (or A100) is needed for LLaMA 70B.

4 Setup

We developed KurTail using the Hugging Face library (Wolf et al., 2019) integrated with the PyTorch framework (Paszke et al., 2019) and for evaluation we used EleutherAI evaluation framework (Gao et al., 2024b). For learning the transformation, we used 512 calibration samples for all models, except Mixteral and LLAMA 70B for which we use 256 calibration sample from the WikiText (Merity et al., 2016a) training set, each with a sequence length of 2048. For large models, we used less samples since they have more layers for which we can store the activations. For storing the activations we used layer-wise inference to reduce the GPU memory requirement. For optimizing the rotation, we use Caley Adam

Table 1: Comparison of different quantization methods across various models. All the results are for 4 bit quantization for Weight/Activation/KV-cache. Weights are quantized using GPTQ.

Method	Llama-2-7b			Llama-2-13b			Llama-3-8b		
	Wiki (↓)	**0-shot (↑)**	**MMLU (↑)**	**Wiki (↓)**	**0-shot (↑)**	**MMLU (↑)**	**Wiki (↓)**	**0-shot (↑)**	**MMLU (↑)**
16-bit	5.5	64.1	42.1	4.9	66.5	52.7	6.1	67.2	63.2
GPTQ	9600.0s	38.9	23.8	3120.0	33.8	24.8	166.3	39.8	23.3
QuaRot	6.2	60.6	32.3	5.4	64.7	46.83	8.50	60.1	47.4
SpinQuant	6.0	61.0	34.8	5.2	64.8	47.8	7.4	63.8	56.2
Kurtail	**5.9**	**61.3**	**32.9**	**5.2**	**65.2**	**49.1**	**7.2**	**64.6**	**57.3**
Method	**Llama-3-70b**			**Llama-3.2-1b**			**Llama-3.2-3b**		
	Wiki (↓)	**0-shot (↑)**	**MMLU (↑)**	**Wiki (↓)**	**0-shot (↑)**	**MMLU (↑)**	**Wiki (↓)**	**0-shot (↑)**	**MMLU (↑)**
16-bit	2.8	73.1	76.3	9.75	54.9	37.9	7.8	62.7	54.8
GPTQ	452.7	45.5	23.2	108.9	38.0	24.9	178.3	40.3	24.8
QuaRot	6.19	65.1	62.9	17.4	49.0	23.8	10.1	56.1	42.0
SpinQuant	6.2	65.7	59.4	13.6	48.8	25.6	9.2	57.9	44.2
Kurtail	**4.2**	**70.7**	**73.1**	**12.9**	**50.1**	**27.2**	**9.0**	**59.0**	**47.8**

(Li et al., 2020) optimizer to find the rotation.

For quantizing the activation, we used per-token dynamic symmetric quantization, where a single scale was applied to each row, and all values were clipped using a quantile of 0.98 in all experiments. For the KV-caches, we employed asymmetric quantization. For the Weight quantization, we use round-to-nearest (RTN), and GPTQ (Frantar et al., 2022), using per-column (or per-channel) symmetric quantization. For GPTQ quantization, we uses 128 calibration samples from the WikiText, each with a sequence length of 2048. Learning the transformation and Transforming LLAMA3-70B with KurTail on an NVIDIA H100 GPU took around one hour which compare the SpinQuant it uses significantly less memory (4 A100 GPU and 2 hours).

Models We evaluate KurTail on the LLAMA-2 (Touvron et al., 2023), LLAMA-3 (Dubey et al., 2024), Phi-model family (Abdin et al., 2024) on both language generation and zero-shot tasks. We further also target the mixture of experts model Mixtral (Jiang et al., 2024).

Inference Speed-up. KurTail's contribution focuses on a novel approach to learning the rotation and given the architectural similarity with SpinQuant and Quarot, we did not re-implement the low-level kernel for 4-bit matrix multiplication, as similar speedup results are expected. All results are based on simulated quantization; however, the real quantization will yield the same downstream performance.

Evaluation Setting. To compare the performance of the model after quantization, we report the perplexity (PPL) score on the WikiText (Merity et al., 2016b) test set. While perplexity is a standard measure of language modeling performance, it may not be sufficient for evaluating the model's effectiveness after quantization. Therefore, we report the result for zero-shot reasoning as well. We assess performance using the lm-evaluation-harness (Gao et al., 2024a), testing the models on eight tasks: BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA(OBQA) (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy, and ARC-Challenge (Boratko et al., 2018) reporting the average performance across all eight tasks (0-shot), we also provide the performance on each task in § C. Additionally, to assess the model on more complex tasks, we benchmark its language comprehension and general understanding using the MMLU benchmark (Hendrycks et al., 2021) and for mathematical reasoning we utilize MathQA (Amini et al., 2019). We report the average performance in Table 1.

5 Results

To evaluate KurTail we focus on 4-bit quantization for weights, activations and KV-cache, which is a challenging bit-width for LLM quantization. Table 1 shows a summary where "0-shot" means the average performance over 8 tasks of common sense reasoning. For weight quantization we used GPTQ (Frantar et al., 2022). We also provide each task performance in Table 2 and and for all model in § C. We report the detailed performance of each tasks in § C. To demonstrate that our method outperforms previous works independently of the weight quantization technique, we alos provide results for

Table 2: Performance comparison of various models with 4 bits W/A/KV-cache quantization in common sense reasoning tasks. All the weight are quantized using GPTQ.

Model	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	PIQA	SIQA	WinoGrande	AVG
Llama-2-7B	Vanilla	46.2	74.5	77.8	76.0	44.2	79.1	46.1	69.1	64.1
	Quarot	41.6	70.6	73.2	72.1	41.2	76.9	44.0	65.2	60.6
	SpinQuant	**43.6**	71.3	**73.8**	73.2	40.4	76.0	44.1	65.4	61.0
	Kurtail	43.1	**72.0**	72.0	**73.2**	**41.2**	**76.6**	**45.6**	**66.8**	**61.3**
Llama-3-8B	Vanilla	53.4	77.8	81.4	79.2	45.0	80.8	47.2	72.6	67.2
	Quarot	42.1	69.0	72.1	71.5	41.2	74.9	44.3	65.5	60.1
	SpinQuant	48.0	75.4	75.8	75.4	**43.8**	77.5	45.0	69.2	63.8
	Kurtail	**48.2**	**75.4**	**79.2**	**76.4**	43.6	**78.4**	**45.8**	**70.0**	**64.6**
Llama-3-70B	Vanilla	65.0	86.6	85.4	85.0	48.2	84.3	50.5	79.9	73.1
	Quarot	53.0	74.8	81.2	77.7	42.0	78.2	45.7	68.4	65.1
	SpinQuant	52.0	77.3	81.7	75.6	43.8	78.8	43.4	72.8	65.7
	Kurtail	**59.2**	**82.7**	**83.9**	**83.3**	**46.6**	**83.5**	**49.7**	**76.6**	**70.7**

round-to-nearest (RTN) in § C. Additionally, to show that our method is effective on LLM families beyond the LLaMA family, we present results on the Phi-3 model in Table 3.

Table 3: Performance on Phi-3-mini-4k-instruct.

Method	Wiki(↓)	0-shot(↑)	MMLU(↑)
16 bit	6.01	0.69	70.75
Quarot	8.46	0.61	56.01
KurTail	**7.13**	**0.66**	**63.61**

Table 4: Performance comparison of different quantization methods for Mixtral-8x7B. All results correspond to 4-bit quantization for weights, activations, and KV-cache. RTN is used for weight quantization.

Method	Mixtral-8x7B		
	Wiki (↓)	0-shot (↑)	MMLU (↑)
16-bit	3.8	71.2	68.8
RTN	909.0	35.4	23.0
QuaRot	8.7	55.7	36.8
Kurtail	**6.5**	**59.4**	**44.8**

For all of the result we have better perplexity in all of the models compared to previous methods. At the same time, our method is significantly better that SpinQuant and QuaRot in downstream tasks. We provide further results for mixture of experts models in Table 4. We also provide results for math reasoning in Table 5.

Experiment on Mixture of Experts Given the growing popularity of the Mixture of Experts (MoE) models, we also explore the idea of applying rotation within the mixture of experts. For this purpose, we utilize Mixtral (Jiang et al., 2024), which employs the exact same attention block. However, for the mixture of experts component, we apply rotation across all the experts. Table 4 presents the results for 4-bit quantization, where we used rounding to the nearest value. In principle, other quantization methods, such as GPTQ, HQQ (Badri and Shaji, 2023), and similar approaches, can also be employed to further enhance performance.

Evaluating Mathematical Reasoning. To explore more complex reasoning tasks, we further evaluate the performance of the quantized model on tasks involving mathematical reasoning in Table 5 by reporting results on the MathQA (Amini et al., 2019) dataset. MathQA is a benchmark designed to test problem-solving and quantitative reasoning abilities. The dataset consists of real-world mathematical problems covering topics such as arithmetic, algebra, probability, and geometry. Each problem is accompanied by a natural language description, multiple-choice answers, and an annotated solution program that outlines the reasoning steps required to reach the correct answer. In Table 5, we compare KurTail with QuaRot, and the results show that KurTail outperforms QuaRot. This additional observation suggests that optimizing the rotations can also enhance performance on math reasoning tasks.

Ablation Study on the Calibration Dataset We also investigate the impact of the calibration dataset on performance. To this end, we modify the calibration data to optimize the rotation using different

Table 5: Comparison of different quantization methods across various for mathematical reasoning on MathQA. All results are reported for 4-bit quantization of W/A/KV-cache. For weight quantization, we use GPTQ.

Model	MathQA Acc (%)		
	16-bit	QuaRot	KurTail
LLaMA-2-7B	28.24	26.70	**26.77**
LLaMA-2-13B	31.76	28.81	**30.35**
LLaMA-2-70B	38.39	33.97	**35.68**
LLaMA-3-8B	40.30	31.36	**34.71**
LLaMA-3-70B	51.79	35.54	**45.76**
LLaMA-3.2-1B	28.94	25.29	**26.00**
LLaMA-3.2-3B	34.67	30.75	**30.52**
Phi-3-mini	39.93	31.89	**34.81**

datasets. Specifically, we conduct experiments using PTB (Marcus et al., 1993), C4 (Raffel et al., 2020), WikiText (Merity et al., 2016b), and Alpaca (Taori et al., 2023). Additionally, we create a combined dataset by sampling equally from all four sources. For each experiment, we sample 512 instances and report the results for Llama-3.2 3B.

Table 6: Performance on different calibration datasets.

Cal Dataset	Wiki(↓)	0-shot(↑)	MMLU(↑)
Quarot	10.1	56.1	42.0
Wikitext-2	**9.0**	59.05	47.76
C4	9.1	59.24	47.75
Alpaca	9.3	59.68	47.34
PTB	9.2	58.60	48.33
Combined	**9.0**	**59.79**	**48.75**

Table 6 presents the findings. Interestingly, all dataset variations outperform the non-training method Quarot. Moreover, we observe lower perplexity on WikiText when using other datasets for calibration. The best performance on the MMLU task is achieved with the PTB dataset, while the best results for common sense reasoning tasks are obtained using the Alpaca dataset. The combined dataset yields the best overall performance across all tasks while it uses the exact same number of samples (512 sentences).

In Table 7, we explore different calibration sample sizes for learning the rotations and their impact on the model's performance in downstream tasks. In this study, we used our combined dataset and the Llama 3.2 3B model. As shown in Table 6, we observe a trend toward improvement as the sample size increases, although performance tends to saturate around a sample size of 512.

Table 7: Effect of different calibration size.

Cal Size	Wiki(↓)	0-shot(↑)	MMLU(↑)
128	9.11	59.24	47.85
256	9.12	58.85	47.47
512	9.09	**59.79**	48.75
1024	**9.08**	59.43	**49.02**

6 Conclusion

We introduced KurTail – a novel technique for learning orthogonal transformations that rotate the activation distribution to address the outlier problem. KurTail effectively reduces quantization sensitivity and minimizes quantization error by tackling important challenges, such as the outlier issue, and overcomes the limitations of previous approaches. Compared to QuaRot, which uses non-learnable rotation, and SpinQuant, which requires substantial computational resources for learning rotations, KurTail provides a more efficient and robust solution. We further provide theoretical insights into why layer-wise optimization yields the same results as end-to-end training, and why orthogonal transformations are a suitable choice of matrix space for learning the transformation. Finally, these results highlight KurTail 's ability to deliver efficiency and high performance across large-scale language models.

Limitations In this work, we only focuses on dynamic per-token quantization for activations, which offers flexibility but does not fully exploit the potential of static tensor-wise quantization. Static quantization, which precomputes scaling factors for improved efficiency, could further optimize inference speed and memory usage. However, it requires careful calibration, which we leave for future work.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly

capable language model locally on your phone, 2024. *URL https://arxiv. org/abs/2404.14219*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

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. Towards end-to-end 4-bit inference on generative large language models. *arXiv preprint arXiv:2310.09259*.

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

Hicham Badri and Appu Shaji. 2023. Half-quadratic quantization of large machine learning models.

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. 2019. Post-training 4-bit quantization of convolution networks for rapid-deployment. *Preprint*, arXiv:1810.05723.

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

Michael Boratko, Harsh Padigela, Deepak Mikkilineni, Pavan Yuvraj, Rajarshi Das, Andrew McCallum, Mihai Chang, Achille Fokoue, Pavan Kapanipathi, Nicholas Mattei, et al. 2018. Arc: A machine reading comprehension dataset for reasoning over science text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1414–1423.

Brian Chmiel, Ron Banner, Gil Shomron, Yury Nahshan, Alex Bronstein, Uri Weiser, et al. 2020. Robust quantization: One model to rule them all. *Advances in neural information processing systems*, 33:5308–5317.

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. *arXiv preprint arXiv:1905.10044*.

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.

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

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*.

Nelson Elhage, Robert Lasenby, and Christopher Olah. 2023. Privileged bases in the transformer residual stream. Transformer Circuits Thread.

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*.

Leo Gao, Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, et al. 2024a. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024b. A framework for few-shot language model evaluation.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Jun Li, Li Fuxin, and Sinisa Todorovic. 2020. Efficient riemannian optimization on the stiefel manifold via the cayley transform. *arXiv preprint arXiv:2002.01113*.

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 on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

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

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

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

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Openbookqa: Fact-based open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 268–277.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms. Accessed: 2025-01-30.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

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, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/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.

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 Proofs

Lemma A.1. *The quantization error after transformation is bounded with* $\left\|X - Q(xT)T^{-1}\right\|_F \leq \frac{c}{2}\sqrt{NM}\left\|\boldsymbol{T}^1\right\|_2\left\|\boldsymbol{T}^{-1}\right\|_2 = \frac{c}{2}\sqrt{NM}\kappa(T)$*, where* T *is the transformation and* c *is a constant depends on* $\boldsymbol{X}$ *and* $\kappa(T)$ *is the condition number of matrix* T*.*

Proof of Lemma 2.3

We aim to bound the quantization error defined as $\left\|\boldsymbol{X} - Q(\boldsymbol{XT})\boldsymbol{T}^{-1}\right\|_F$. To do so, we use the fact that $\boldsymbol{X} = \boldsymbol{XTT}^{-1}$:

$$\begin{aligned}\mathbb{E} &= \left\|\boldsymbol{X} - Q(\boldsymbol{XT})\boldsymbol{T}^{-1}\right\|_F \\ &= \left\|\boldsymbol{XTT}^{-1} - Q(\boldsymbol{XT})\boldsymbol{T}^{-1}\right\|_F.\end{aligned} \tag{5}$$

Applying the sub-multiplicative property of the Frobenius norm (i.e., $\|AB\|_F \leq \|A\|_2\|B\|_F$), we obtain:

$$\begin{aligned}&\left\|\boldsymbol{XTT}^{-1} - Q(\boldsymbol{XT})\boldsymbol{T}^{-1}\right\|_F \leq \\ &\left\|\boldsymbol{XT} - Q(\boldsymbol{XT})\right\|_F \cdot \left\|\boldsymbol{T}^{-1}\right\|_2.\end{aligned} \tag{6}$$

We now focus on bounding the quantization error term $\left\|\boldsymbol{XT} - Q(\boldsymbol{XT})\right\|_F$. Under uniform quantization, each entry of the quantization error is bounded by $\frac{\Delta}{2}$, where $\Delta = \frac{\max_{ij}|(\boldsymbol{XT})_{ij}|}{2^{k-1}-1}$ is the quantization step size. Therefore, the Frobenius norm can be bounded by:

$$\left\|\boldsymbol{XT} - Q(\boldsymbol{XT})\right\|_F \leq \sqrt{NM} \cdot \frac{\Delta}{2}, \tag{7}$$

where N and M are the number of rows and columns of $\boldsymbol{XT}$, respectively.

Combining Eq. 6 and Eq. 7, we get:

$$\left\|\boldsymbol{X} - Q(\boldsymbol{XT})\boldsymbol{T}^{-1}\right\|_F \leq \sqrt{NM} \cdot \frac{\Delta}{2} \cdot \left\|\boldsymbol{T}^{-1}\right\|_2. \tag{8}$$

To bound Δ, we use the fact that the maximum absolute value of elements in $\boldsymbol{XT}$ satisfies:

$$\max_{ij}|(\boldsymbol{XT})_{ij}| \leq \|\boldsymbol{XT}\|_2 \leq \|\boldsymbol{X}\|_2 \cdot \|\boldsymbol{T}\|_2. \tag{9}$$

Substituting this into the expression for Δ, we obtain:

$$\Delta \leq \frac{\|\boldsymbol{X}\|_2 \cdot \|\boldsymbol{T}\|_2}{2^{k-1}-1}.$$

Finally, substituting this into Eq. 8, we conclude:

$$\begin{aligned}&\left\|\boldsymbol{X} - Q(\boldsymbol{XT})\boldsymbol{T}^{-1}\right\|_F \leq \\ &\frac{\|\boldsymbol{X}\|_2}{2^{k-1}-1} \cdot \sqrt{NM} \cdot \|\boldsymbol{T}\|_2 \cdot \left\|\boldsymbol{T}^{-1}\right\|_2 = \\ &\frac{\|\boldsymbol{X}\|_2}{2^{k-1}-1} \cdot \sqrt{NM} \cdot \mathrm{cn}(\boldsymbol{T}),\end{aligned} \tag{10}$$

where $\mathrm{cn}(\boldsymbol{T}) = \|\boldsymbol{T}\|_2 \cdot \left\|\boldsymbol{T}^{-1}\right\|_2$ denotes the condition number of $\boldsymbol{T}$.

Proposition A.2. *Let* $\boldsymbol{H}_1, \boldsymbol{O}_1 = f(\boldsymbol{X}, \boldsymbol{W}_1, \boldsymbol{R}_1)$*, and* $\boldsymbol{H}_2, \boldsymbol{O}_2 = g(\boldsymbol{O}_1; \boldsymbol{W}_2; \boldsymbol{R}_2)$ *where* f, g *are parameterized by* $\boldsymbol{W}_1, \boldsymbol{R}_1$ *and* $\boldsymbol{W}_2, \boldsymbol{R}_2$*. Given functional invariance of* f *and* g*, i.e.* $\boldsymbol{O}_1 = \boldsymbol{O}_1'$ *for any* $\boldsymbol{O}_1', \boldsymbol{H}_1' = f(\boldsymbol{X}, \boldsymbol{W}_1, \boldsymbol{R}_1')$ *and any orthogonal* $\boldsymbol{R}_1'$ *(and similarly for* g*), and given that the total loss is* $\mathcal{L}(\boldsymbol{R}_1, \boldsymbol{R}_2) = \mathcal{L}_1(\boldsymbol{H}_1) + \mathcal{L}_2(\boldsymbol{H}_2)$*, the independent minimization of each loss results in the same optimum as end-to-end:* $\arg\min_{\boldsymbol{R}_1,\boldsymbol{R}_2} \mathcal{L}(\boldsymbol{R}_1, \boldsymbol{R}_2) = (\arg\min_{\boldsymbol{R}_1} \mathcal{L}_1(\boldsymbol{R}_1),\ \arg\min_{\boldsymbol{R}_2} \mathcal{L}_2(f; \boldsymbol{R}_2))$*, even though* $\boldsymbol{H}_2$ *implicitly depends on* $\boldsymbol{O}_1$*.*

Proof of Proposition 2.5

The proof is also intuitive since, condition 1 implies that the arguments to L_i are constants w.r.t. R_j. Condition 2 implies L_i depends only on R_i. Thus, $\mathcal{L}(R_1, R_2) = \mathcal{L}_1(R_1) + \mathcal{L}_2(f; R_2)$, which is separable.

B Evaluation of KurTail on Channel Outliers.

To demonstrate that the learned rotation by KurTail reduces the degree of tailedness in the distribution, we visualize the inputs of multi-head self-attention (MHSA) and feed-forward network (FFN) blocks of layer 15 in Llama3-8B. In Fig. 2, we compare the input distribution once without rotation and once with KurTail learned rotation. Additionally, we highlight the maximum value for each token with a gray surface above each token. As shown, KurTail effectively mitigates outliers in activation quantization.

In dynamic per-token quantization, the maximum value of a token's vector plays a critical role in determining the quantization step size and range. Larger maximum values increase the quantization range, which results in larger quantization steps and greater precision loss. Alternatively, reducing the maximum value allows for smaller quantization steps, which result in more efficient representation of token values with minimal degradation of information. Therefore, lowering the maximum values

across tokens is directly connected to overall quantization error and model performance. To evaluate how well different methods achieve this goal, we measure the success rate of our proposed method, KurTail , compared to its un-rotated counterpart (baseline vector) and an alternative rotation method, QuaRot. A "success" is defined as a case where the maximum value of a token's vector after applying a benchmark rotation method (KurTail or QuaRot) is smaller than that of the baseline vector. The success rate is defined as the percentage of tokens where the benchmarked rotated version achieves this reduction. In Table 8, we present the average success rates for LLAMA3-8B. KurTail consistently produces smaller maximum values across all layers, samples, and tokens, achieving a higher success rate compared to the baseline vector in nearly all cases. Additionally, it outperforms QuaRot in approximately 63.29% in MSHA, 62.99% in FFN on average.

Table 8: The success rate of benchmark over baseline.

	Baseline	Benchmark	Success Rate (%)
MHSA	Vanilla	KurTail	99.74%
	Vanilla	QuaRot	99.43%
	QuaRot	KurTail	63.29%
FFN	Vanilla	KurTail	99.96%
	Vanilla	QuaRot	99.96%
	QuaRot	KurTail	62.99%

C Further Evaluation

In this section, we provide a more detailed evaluation of all tasks and more models. We present results for 4-bit quantization of weights, activations, and the KV-cache. Table 9 reports the performance of each MMLU task under 4-bit quantization for weights, activations, and the KV-cache. We use the GPTQ quantization algorithm for weight quantization in this experiment. Similarly, using the same setup, we evaluate common-sense reasoning tasks, as shown in Table 10. Finally, we report the performance of common-sense reasoning tasks using RTN quantization for weights in Table 11.

Table 9: Performance comparison of different models using various methods across different domains.

Model	Method	Human	Other	STEM	S-Sci	AVG
Llama-2-7B	Vanilla	39.8	47.3	34.2	47.3	42.1
	Quarot	31.1	35.7	29.9	34.1	32.7
	SpinQuant	33.9	38.5	29.5	37.5	34.8
	Kurtail	32.3	35.0	29.8	34.4	32.9
Llama-2-13B	Vanilla	47.9	59.4	42.3	61.2	52.7
	Quarot	42.7	52.3	38.2	54.1	46.8
	SpinQuant	43.5	53.1	39.1	55.4	47.8
	Kurtail	45.3	54.0	40.4	56.6	49.1
Llama-3-8B	Vanilla	55.0	70.8	53.7	73.2	63.2
	Quarot	42.1	52.9	39.8	54.9	47.4
	SpinQuant	49.8	63.3	46.8	65.0	56.2
	Kurtail	50.2	64.5	49.1	65.6	57.3
Llama-3-70B	Vanilla	67.7	81.5	69.2	86.7	76.3
	Quarot	55.3	68.5	53.7	74.1	62.9
	SpinQuant	50.7	67.0	51.9	68.1	59.4
	Kurtail	65.2	79.1	63.9	84.2	73.1
Llama-3.2-1B	Vanilla	35.3	41.3	33.9	41.3	38.0
	Quarot	25.4	26.9	24.4	25.4	25.5
	SpinQuant	25.4	27.6	24.2	25.3	25.6
	Kurtail	26.5	28.8	26.0	27.3	27.2
Llama-3.2-3B	Vanilla	49.0	63.1	45.5	62.9	55.1
	Quarot	38.5	47.3	35.3	46.7	42.0
	SpinQuant	37.0	49.4	39.9	50.5	44.2
	Kurtail	44.8	53.4	39.5	53.4	47.8

Table 10: Performance comparison of various models with 4 bits W/A/KV-cache quantization in common sense reasoning tasks. All the weight are quantized using GPTQ.

Model	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	PIQA	SIQA	WinoGrande	AVG
Llama-2-7B	Vanilla	46.2	74.5	77.8	76.0	44.2	79.1	46.1	69.1	64.1
	Quarot	41.6	70.6	73.2	72.1	41.2	76.9	44.0	65.2	60.6
	SpinQuant	43.6	71.3	73.8	73.2	40.4	76.0	44.1	65.4	61.0
	Kurtail	43.1	72.0	72.0	73.2	41.2	76.6	45.6	66.8	61.3
Llama-2-13B	Vanilla	49.2	77.5	80.6	79.4	45.2	80.5	47.4	72.1	66.5
	Quarot	47.3	73.9	77.8	76.6	44.4	78.7	44.1	69.8	64.1
	SpinQuant	49.0	76.3	78.2	77.1	42.8	79.3	46.3	69.5	64.8
	Kurtail	48.1	75.4	79.7	77.4	45.0	79.0	45.6	71.2	65.2
Llama-3-8B	Vanilla	53.4	77.8	81.4	79.2	45.0	80.8	47.2	72.6	67.2
	Quarot	42.1	69.0	72.1	71.5	41.2	74.9	44.3	65.5	60.1
	SpinQuant	48.0	75.4	75.8	75.4	43.8	77.5	45.0	69.2	63.8
	Kurtail	48.2	75.4	79.2	76.4	43.6	78.4	45.8	70.0	64.6
Llama-3-70B	Vanilla	65.0	86.6	85.4	85.0	48.2	84.3	50.5	79.9	73.1
	Quarot	53.0	74.8	81.2	77.7	42.0	78.2	45.7	68.4	65.1
	SpinQuant	52.0	77.3	81.7	75.6	43.8	78.8	43.4	72.8	65.7
	Kurtail	59.2	82.7	83.9	83.3	46.6	83.5	49.7	76.6	70.7
Llama-3.2-1B	Vanilla	36.2	60.4	63.9	63.6	37.2	74.6	43.0	60.5	54.9
	Quarot	30.0	51.4	59.1	54.0	34.2	66.7	39.6	57.1	49.0
	SpinQuant	32.3	51.8	59.3	55.4	30.4	67.7	38.6	54.7	48.8
	Kurtail	31.1	52.9	60.7	56.4	36.4	68.6	40.5	54.3	50.1
Llama-3.2-3B	Vanilla	46.0	71.7	73.2	73.6	43.0	77.5	47.0	69.7	62.7
	Quarot	38.6	59.0	65.9	66.5	35.8	74.4	43.1	65.2	56.1
	SpinQuant	38.9	64.8	68.0	69.1	39.4	74.9	45.1	62.9	57.9
	Kurtail	42.2	66.7	69.8	68.8	39.8	75.6	44.8	64.6	59.0

Table 11: Performance comparison of various models with 4 bits W/A/KV-cache quantization in common sense reasoning tasks. All the weights are quantized using RTN.

Model	Method	ARC-C	ARC-E	BoolQ	HellaSwag	OBQA	PIQA	SIQA	Winogrande	AVG
Llama-2-7B	Vanilla	46.2	74.5	77.8	76.0	44.2	79.1	46.1	69.1	64.1
	Quarot	35.2	62.4	69.0	62.6	33.4	71.7	40.9	60.2	54.4
	Kurtail	39.0	64.9	69.8	64.7	39.2	74.1	42.1	62.2	57.0
Llama-2-13B	Vanilla	49.2	77.5	80.6	79.4	45.2	80.5	47.4	72.1	66.5
	Quarot	41.4	68.2	73.2	71.2	41.6	76.3	41.1	66.1	59.9
	Kurtail	44.2	70.3	74.7	72.5	40.4	77.5	45.9	70.2	62.0
Llama-2-70B	Vanilla	57.4	81.1	83.8	83.8	48.8	82.8	49.2	78.0	70.6
	Quarot	50.5	76.8	80.0	78.4	44.0	79.9	46.0	72.9	66.1
	Kurtail	51.3	76.6	80.9	81.0	46.4	81.7	46.8	76.2	67.6
Llama-3-8B	Vanilla	53.4	77.8	81.4	79.2	45.0	80.8	47.2	72.6	67.2
	Quarot	31.1	51.6	55.7	62.0	31.6	66.3	40.1	59.0	49.7
	Kurtail	38.1	61.1	72.5	69.3	36.8	72.9	41.9	66.1	57.3
Llama-3-70B	Vanilla	65.0	86.6	85.4	85.0	48.2	84.3	50.5	80.0	73.1
	Quarot	20.6	31.3	58.5	28.4	25.4	55.0	33.2	50.7	37.9
	Kurtail	23.0	37.8	48.5	33.9	29.8	61.8	36.6	51.6	40.4
Llama-3.2-1B	Vanilla	36.2	60.4	63.9	63.6	37.2	74.6	43.0	60.5	54.9
	Quarot	27.4	33.9	39.1	36.2	30.0	56.9	34.7	53.0	38.9
	Kurtail	28.7	37.2	38.8	42.9	31.6	60.0	35.7	57.5	41.5
Llama-3.2-3B	Vanilla	46.0	71.7	73.2	73.6	43.0	77.5	47.0	69.7	62.7
	Quarot	33.1	50.3	41.8	56.3	31.8	67.8	39.8	56.8	47.2
	Kurtail	37.4	56.6	48.0	62.1	36.6	71.3	40.5	60.4	51.6