Gradual Binary Search and Dimension Expansion : A general method for activation quantization in LLMs

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) have become pivotal in artificial intelligence, demonstrating strong capabilities in reasoning, understanding, and generating data. However, their deployment on edge devices is hindered by their substantial size, often reaching several billion parameters. Quantization is a widely used method to reduce memory usage and inference time, however LLMs present unique challenges due to the prevalence of outliers in their activations. In this work, we leverage the theoretical advantages of Hadamard matrices over random rotation matrices to push the boundaries of quantization in LLMs. We demonstrate that Hadamard matrices are more effective in reducing outliers, which are a significant obstacle in achieving low-bit quantization. Our method based on a gradual binary search enables 3-bit quantization for weights, activations, and key-value (KV) caches, resulting in a 40% increase in accuracy on common benchmarks compared to SoTA methods. We extend the use of rotation matrices to support non-power-of-2 embedding dimensions, similar to the Qwen architecture, by employing the Paley's algorithm. Our experimental results on multiple models family like Mistral, LLaMA, and Qwen demonstrate the effectiveness of our approach, outperforming existing methods and enabling practical 3-bit quantization.

1 Introduction

Large Language Models (LLMs) have become a central component of artificial intelligence due to their strong capabilities in reasoning, understanding, and generating data. These impressive capabilities are attributed to the quality of the data used during training, the model architecture, and the size of the model, which often reaches several billion parameters. This size limitation restricts their deployment on edge devices. Quantization is a widely used method to reduce memory usage and inference time (Gholami et al. (2021); Guo (2018)), but the challenges differ compared to those faced with Convolutional Neural Networks (CNNs) (Esser et al. (2020); Xiao et al. (2023)).

Weights are relatively easy to quantize for both CNNs and LLMs and can often achieve ternary quantization without significant loss of accuracy (Ma et al. (2024); Zhu et al. (2017)). However, activations behave differently in transformer architectures (Nrusimha et al. (2024)). The presence of outliers in activations makes conventional quantization (symmetric uniform) very challenging, hindering our ability to achieve 4-bit quantization. LLMs are known to produce spikes in its layers and for some tokens that can be handled separately or diffused in the tensor (Dettmers et al.; Xiao et al. (2023)).

One very promising approach to overcome this limitation is to use rotation matrices to redistribute weights and activation values, thereby minimizing the impact of outliers (Liu et al. (2024); Ashkboos et al. (2024b)). Additionally, methods such as prefix tokens have shown very interesting results in managing outliers in LLMs (Chen et al. (2024); Son et al. (2024)).

In this work, we leverage results on rotation matrices to push the boundaries further and enable 3-bit Weights, Activations, KV cache (WAKV) quantization by employing a binary search. We extend this method to a more general approach capable of handling non-power-of-2 embedding dimensions, similar to Qwen. Our main contributions are:

- A theoretical demonstration that Hadamard matrices are more effective in reducing multiple outliers than rotation matrices drawn on the unit sphere.
- 3-bit quantization for weights, activations, and KV cache, resulting in a 40% increase in accuracy on common benchmarks using a gradual binary search.
- Extension of rotation matrices to support non-power-of-2 embedding dimensions using the Paley's algorithm.
- The introduction of dimension expansion to build a more general rotation pipeline allowing architectures like Qwen to work with rotations.

2 Related Works

2.1 Quantization

Quantizing models involves reducing the number of bits required to store and compute model activations. This process is crucial for deploying LLMs on resource-constrained devices. To achieve this, we define a scaling factor that determines the distance between quantization bins and the range of values to be compressed.

For symmetric uniform quantization, we apply a rounding function to a scaled distribution:

$$\hat{X} = \text{round}\left(\frac{X}{\Delta}\right)\Delta,\ \Delta = \frac{\max|X|}{2^b - 1}$$

where Δ is the scaling factor, b is the bitwidth, and $\max|X|$ is the maximum absolute value of the distribution, preserving extreme values for activations.

Such quantization can be applied per-token, where each token has a different scaling factor, or per-tensor, where a single scaling factor is used for each activation tensor (Gholami et al. (2021); Guo (2018)). Per-token quantization is more challenging to implement efficiently in practice compared to per-tensor quantization but results in better quantization performances. Scaling factors can be static during inference, based on statistics computed on a subset of the dataset, or dynamic, recomputed at each step.

Quantization can lead to a significant drop in performance when applied post-training (PTQ) (Yang et al. (2023)). To mitigate this, some methods adapt weights to the noise introduced during a training phase (QAT) (Lin et al. (2019); Défossez et al. (2022)). Typically, for LLMs, only linear layers are quantized, as they account for most of the computation cost, while normalization layers, matrix multiplications, and the softmax function within the attention block are left unquantized.

2.2 Outliers

Quantizing LLM weights is relatively straightforward and does not require extensive efforts to achieve. Techniques like GPTQ (Frantar et al. (2023)) enables 8-bit quantization without retraining, preserving model accuracy. Some QAT methods can even push the boundaries to 1-bit quantization, as seen in BitNet (Wang et al. (2023)) or ternary quantization (Ma et al. (2024)).

However, LLMs present unique challenges due to the prevalence of extreme high values in their activations (Wei et al. (2023); Nrusimha et al. (2024); Huang et al. (2024); Lin et al.). The scaling factor, which is directly tied to the maximum absolute value, often causes most of the distribution to be rounded to zero, leading to performance degradation. To address this, techniques like LLM.int8() (Dettmers et al.) cluster these outliers and quantize them separately from the main distribution.

Alternative methods, such as SmoothQuant (Xiao et al. (2023)), shift the quantization challenge from activations to weights by introducing a scaling parameter between them. Other approaches attempt to relocate these spikes into "sink tokens" before quantization (Son et al. (2024)). Some research focuses on understanding the upstream causes of these spikes during the learning process to limit their impact post-training

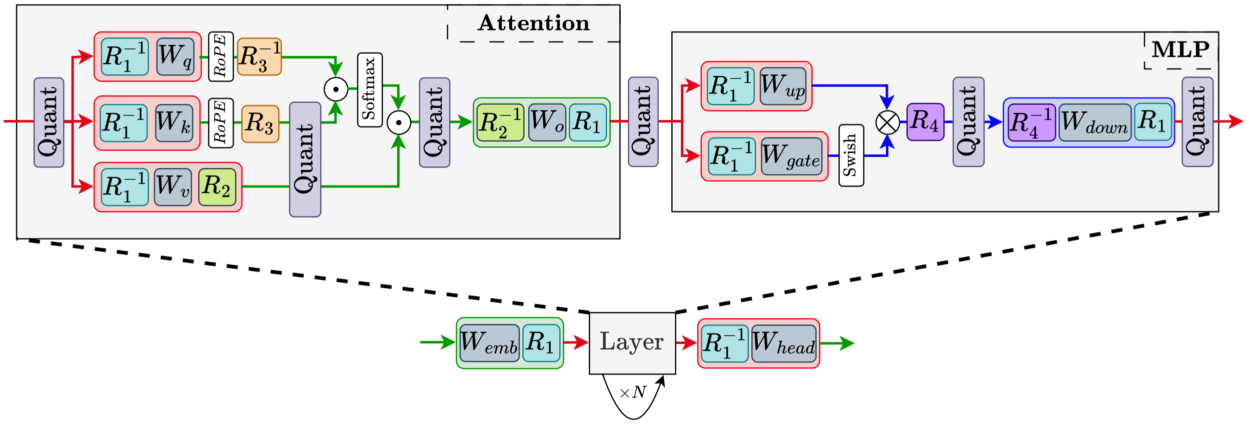


Figure 1: Architecture's pipeline with rotation matrices R_1, R_2, R_3, R_4 and dimension expansion. Red lines represent expanded tokens by d dimensions, green lines represents non expanded tokens and blue lines represent expanded tokens by d'. Weights tensors are also expanded in this pipeline, QKV and Head projections have their input expanded by d and their output remains unchanged whereas Out and Embedding projections only have their output dimension expanded by d. For Up and Gate the input is expanded by d and the output by d'. Finally Down proj has its input expanded by d' and output by d.

(Nrusimha et al. (2024)). Additionally, efforts are made to better locate these outliers by visualizing the layers, dimensions, and tokens that may be their source (Maisonnave et al. (2025)).

2.3 Rotation Matrices

2.3.1 Random orthogonal matrices

Rotation matrices play a pivotal role in various applications, including signal processing, computer vision, and machine learning. These matrices are orthogonal and invertible by their transpose, meaning they preserve the length of vectors and the angles between them. In the context of quantization, rotation matrices can be used to decorrelate and redistribute the energy of model activations (Ashkboos et al. (2024a;b); Chee et al.), making them more amenable to quantization. The idea is to apply orthogonal matrices before quantization to flatten the distribution and then recover the tensor by applying its inverse (see Figure 1). Part of this process can be pre-computed and fused with weights and the rest needs to be done at inference (Ashkboos et al. (2024a)).

However, the effectiveness of rotation matrices depends on the specific matrix used. Randomly drawn orthogonal rotation matrices can introduce noise and reduce the overall performance of the model. To mitigate this, some methods adapt the rotation matrices during a training phase to better align with the model's weights and activations (Liu et al. (2024)).

In practice, rotation matrices are often used in conjunction with other quantization techniques, such as GPTQ. This combination allows for more robust and efficient quantization of large language models, enabling their deployment on resource-constrained devices.

2.3.2 Hadamard matrices

Hadamard matrices are another powerful tool in the quantization arsenal. These matrices are orthogonal matrices and all their entries are either +1 or -1 making them very efficient to compute (eq 1). Hadamard matrices have been extensively used in signal processing, error-correcting codes (Horadam (2012)), and more recently, in the quantization of neural networks (Ashkboos et al. (2024a;b)).

One of the key advantages of Hadamard matrices is their ability to decorrelate the activations of a model. By applying a Hadamard matrix, the activations are transformed into a new basis where the correlations between different dimensions are minimized. This decorrelation property is particularly useful in reducing the impact of outliers, as the extreme values are spread out across multiple dimensions.

Hadamard matrices of order 2^n can be constructed recursively using the Fast Hadamard Transform (FHT) method: For $n \geq 1$, construct the $2^{n+1} \times 2^{n+1}$ Hadamard matrix $H_{2^{n+1}}$ using the $2^n \times 2^n$ Hadamard matrix H_{2^n} as follows:

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}, \quad H_{2^{n+1}} = \begin{pmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{pmatrix} \tag{1}$$

This method is highly efficient for generating Hadamard matrices and can be applied in real-time. In summary, both rotation matrices and Hadamard matrices are essential for the quantization of large language models. However, Hadamard matrices offer several advantages: they can be generated more efficiently, their structure of containing only 1 and -1 makes them highly efficient for matrix multiplication, and they are known to handle outliers in activations more effectively (Liu et al. (2024)). In the following sections, we will theoretically demonstrate that Hadamard matrices are more effective than random rotation matrices drawn from the unit sphere in reducing the amplitude of outliers.

2.3.3 Paley algorithm

To generate other dimensions n for Hadamard matrix we can use known small matrices and apply power of 2 algorithm as used in QuaRot (Ashkboos et al. (2024b)) but it can be limiting and doesn't cover a lot of values. To overcome this issue we can use the Paley's Algorithm that generate a Hadamard matrix $n \times n$ if $n-1$ is a prime number and $n-1 \equiv 3 \pmod 4$. This algorithm is described below (Algorithm 2) and needs to generate Legendre symbols $\left(\frac{a}{p}\right)$ which take any integer number a and prime number p to produce a value in $\{-1, 0, 1\}$ as below :

- If a is a quadratic residue modulo p, then there exists an integer x such that $x^2 \equiv a \pmod p$. In this case, $\left(\frac{a}{p}\right) = 1$.
- If a is a quadratic non-residue modulo p, then there is no integer x such that $x^2 \equiv a \pmod p$. In this case, $\left(\frac{a}{p}\right) = -1$.
- If $a \equiv 0 \pmod p$, then $\left(\frac{a}{p}\right) = 0$.

Generating Legendre symbols can be time-consuming, especially for high-dimensional matrices. However, in the following sections, we will use this algorithm to generate non-power-of-2 Hadamard matrices and fuse them with the weights, so we only need to compute the Legendre symbols once.

3 Analysis and theoretical demonstrations

3.1 Clipping Ratio

To perform quantization we can play on several parameters to improve the effectiveness of the process, for example in LSQ (Esser et al. (2020)) they optimise the scaling factor trough training, or FracBits (Yang & Jin (2021)) which tries to find the best precision for every layer. Other works highlighted the importance of the clipping ratio like PACT (Choi et al. (2018)) where the optimization is done during training. Some others apply a Grid Search (Chen et al. (2024)) to find the best configuration particularly useful for LLMs where training or fine tuning can be very time consuming.

Clipping ratios are essential for managing outliers, as they establish the balance between maintaining high precision for small values and preserving a maximum value close to its original. However, the model exhibits significant variability in how quantization responds to changes in the clipping ratio. While some projections can tolerate very low clipping ratios, others experience a substantial accuracy drop with even slight adjustments (see Appendix C). Therefore, to effectively manage this variability, a tailored clipping ratio must be determined for each projection.

Previous studies have shown that quantization error is not always the best metric to guide the optimization process for quantization parameters (Maisonnave et al.). Specifically, at very low precision levels, such as 4 or 3 bits, the set of quantized weights deviates significantly from the optimized configuration obtained during training. Attempting to recover this configuration using quantization error often results in an ineffective set of weights. To address this issue, we can use perplexity as an objective function. Perplexity provides a more accurate representation of model performance and is computationally efficient, as it is based on Cross Entropy Loss, which is frequently used during training for its smoothness.

3.2 Hadamard Matrices reduce outliers more

Experimentally, it is observed that Hadamard matrices tend to reduce better the amplitude of outliers present in the layers of LLMs, which directly impacts the performance of these models. However, the question of why such a phenomenon occurs has remained open from a theoretical perspective. We now provide an answer to this question.

Definition 3.1. *We define μ a the function that compute the maximum absolute value of a matrix product:*

$$\mu(Mx) = \max_{1 \leq i \leq n} |(Mx)_i|$$

Theorem 3.1 (Hadamard reduction)**.** *$\forall x \in \mathbb{R}^n$ containing k outliers of equal amplitude, i.e., $x = \epsilon + \sum_{j=1}^{k} ce_{p_j}$ with $c >> ||\epsilon||$, e_i denotes the canonical basis vector at position i and p_1, ..., p_k are distinct, we have*

$$\frac{\mu(Hx)}{\mu(Qx)} \sim \sqrt{\frac{k}{2\log n}}$$

with H a Hadamard matrix belonging to $\mathbb{R}^{n\times n}$ and Q a rotation matrix drawn randomly on the unit sphere $\mathcal{S}^{n-1} = \{x \in \mathbb{R}^n : ||x||_2 = 1\}$.

To demonstrate Theorem 3.1, we can calculate the two terms of the fraction and thus show its behavior asymptotically.

Lemma 3.1 (Hadamard incoherence)**.** *For H a Hadamard matrix belonging to $\mathbb{R}^{n\times n}$ and $x = \epsilon + \sum_{j=1}^{k} ce_{p_j}$ with $c >> ||\epsilon||$ a vector containing k outliers of equal amplitude, we have :*

$$\mu(Hx) \sim \frac{kc}{\sqrt{n}}$$

Lemma 3.2 (Rotation incoherence)**.** *For Q a rotation matrix drawn randomly on the unit sphere $\mathcal{S}^{n-1} = \{x \in \mathbb{R}^n : ||x||_2 = 1\}$ and $x = \epsilon + \sum_{j=1}^{k} ce_{p_j}$ with $c >> ||\epsilon||$ a vector containing k outliers of equal amplitude, we have*

$$\mu(Qx) \sim c\sqrt{\frac{2k\log n}{n}}$$

We can prove in Lemma 3.1 and Lemma 3.2 that the reduction of outliers with a Hadamard matrix is of order $O(\sqrt{\frac{k}{n}})$ and $O(\sqrt{\frac{2k\log n}{n}})$ for a random orthogonal matrix (demonstrations are done in Appendix A). These results prove Theorem 3.1 and also show the close link between reduction and the dimension of embeddings in LLMs. The higher the dimension is the stronger the reduction will be.

Algorithm 1 Gradual Binary Search

```
Require: A model $M$, a dataset $D$, a threshold $\epsilon$
Ensure: A list L of clipping ratios                                   ▷ + operand on L means concatenation
$n \leftarrow$ number of projections in M
$L \leftarrow [\,]$
for $i \leftarrow 1$ to $n - 1$ do
    $a \leftarrow 0$
    $b \leftarrow 1$
    $m \leftarrow (a + b)/2$                                         ▷ We keep track of the middle element
    $M \leftarrow$ quantize_proj$(M, i)$                              ▷ Quantize projection $i$ of model $M$
    $f_m =$ evaluate$(M, D, L + m)$               ▷ Evaluate model $M$ on dataset $D$ with clipping ratios L
    iteration $\leftarrow 0$
    while $b - a > \epsilon$ do                                      ▷ We iterate until we converge
        if iteration is even then                     ▷ Allows to use only one loop for binary search
            $x \leftarrow (a + m)/2$
        else
            $x \leftarrow (b + m)/2$
        end if
        $f_x \leftarrow$ evaluate$(M, D, L + x)$                    ▷ Evaluate model on a new clipping ratio
        if $f_x < f_m$ then                     ▷ If we improve PPL (the lower the better) we keep it
            if $x < m$ then          ▷ If the target is less than the middle, search the left half
                $b \leftarrow m$
            else                                                     ▷ If not , search the right half
                $a \leftarrow m$
            end if
            $m, f_m \leftarrow x, f_x$
        else
            if $x < m$ then
                $a \leftarrow x$
            else
                $b \leftarrow x$
            end if
        end if
        iteration $\leftarrow$ iteration $+ 1$
    end while
    $L = L + m$                                                     ▷ Add new element to the list
end for
return $L$
```

4 Method

4.1 Gradual Binary Search

In Section 3.1, we emphasize the importance of the clipping ratio parameter and its significant impact on model performance. We stress the need to optimize each projection with its own clipping ratio for best results. Our primary contribution is an algorithm that determines the optimal clipping ratio for each quantizer using a binary search (Algorithm 1). To drive the binary search, we minimize perplexity across various clipping ratios, assuming a single minimum and a convex landscape. Additionally, we quantize our model gradually: first, we quantize and optimize the initial linear projection while keeping the rest in FP16, then use the obtained parameters to quantize and optimize the next projection, and so on. This process is discussed in Appendix C where we experimentally show the necessity to optimize gradually the clipping ratio.

4.2 Increasing dimensions

Lemma 4.1 (Expanding limit)**.** *For a matrix product AB with $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ in b bits and $A'B'$ with $A' \in \mathbb{R}^{m\times(n+d)}$ and $B' \in \mathbb{R}^{(n+d)\times p}$ in b' bits we must have $d \leq \frac{n(b-b')}{b'}$ so that BitOps$(A'B') \leq$ BitOps(AB), with m, n, p, b, $b' \in \mathbb{N}$ and $b' \leq b$*

One important limitation of QuaRot's implementation of rotation matrices in LLMs is the necessity to have embeddings in a power of 2 dimension which can be very limiting in some architectures like Qwen2.5-7B. In the MLP the dimension is 18944 which can be decomposed as $18944 = 128 \times 148$ but 148 is not a known dimension for an Hadamard matrix even for the Paley algorithm introduced in the paper. To overcome this problem we increase manually the dimension of embedding by adding zeros in the weights (independently developed in Franco et al. (2025)) to reach a dimension suitable to generate a Hadamard matrix with the Paley's algorithm 2. Then we save the matrix product of weights padded with 0s and the Hadamard matrix as our new weights (see figure 1 and Eq 2 for an example in dimension 4). The primary goal is to create a more versatile pipeline compatible with any architecture but it also enhance performance through increased dimensionality.

$$W \leftarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} a & b \\ c & d \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{2}$$

Indeed theorem 3.1 ensures that increasing the dimension helps reduce the impact of outliers in any tensor. Consequently, by adding zeros to the weight tensors, we also improve the effectiveness of quantization. The intuition behind it is that by adding more dimensions in our tensors we create more space to store information and especially outliers which will be sliced in more parts and recovered better after quantization. This process increases the model size and computational cost, necessitating a trade-off to achieve better accuracy without a significant increase in computational requirements.

Lemma 4.1 shows the threshold after which the increase in dimensionality is worse than just quantizing with one more bit. For example with a LLaMA3-8B which has embeddings in 4096 dimensions we are only allowed to increase to $d = 1366$ dimensions in 3 bits before reaching the computational cost in 4 bits.

5 Experiments

5.1 Setup

We conduct our experiment based on the the code of QuaRot which performs per-token quantization for activations and GPTQ for weights. We also quantize KV caches using asymmetric quantization with a group size of 128. We compare our results on several metrics : perplexity (PPL) on WikiText2, and 6 benchmarks : PIQA, hellaswag (HS), arc-easy (ARC-E), arc-challenge (ARC-C), winogrande (WINO) and lambada, we also compute the average value (AVG) of these 6 benchmarks. We performs ours experiments in 4 and 3 bits quantization on 6 different models from the Mistral library, LLaMA architecture and Qwen. We used only one GPU A100 to perform quantization and Gradual Binary Search (GBS) with 10% of the train set of WikiText2 for 3 days for the biggest models. We were able to adjust some optimisations parameters to reduce this computation time to 12h without significant loss of accuracy (see Appendix E).

5.2 Results

5.2.1 Gradual Binary Search performances

Table 1 and Table 2 shows the results in 4 and 3 bits quantization on the perplexity and 6 benchmarks. In 4 bits our method GBS clearly outperforms previous methods for all models improving up to almost 6% for LLaMA3-8B, 5% on Qwen2.5 1.5B Instruct, 4% on Mistral 7B and 3% on Mistral 7B Instruct.

In 3 bits GBS made activation quantization possible with an increase of accuracy reaching 40% for Mistral 7B (Table 2). All other models have been greatly affected by GBS reducing the gap with 4 bits quantization. PPL is also significantly impacted by GBS reducing by a factor of 100 in the case of LLaMA3-8B. We now have a method reaching decent performances in 3 bits like with Mistral 7B Instruct which is only 10% less than FP16 and reach 61.32% accuracy on our benchmarks.

GBS appears to be highly effective in enhancing quantization performance, supporting our hypothesis that optimizing Perplexity via binary search is preferable to minimizing quantization error. We assumed a single minimum and a convex function, allowing us to leverage binary search while relying on the smoothness of CrossEntropy—an assumption that appears to hold true. Perplexity emerges as a strong objective for guiding our optimization, as it correlates well with improved benchmark performance (see Appendix D for more details).

Table 1: Results in 4-bit WAKV quantization on seven benchmarks (perplexity, PIQA, HellaSwag, ARC-Easy, ARC-Challenge, Winogrande, and LAMBADA) and report average success rates. Our GBS method outperforms QuaRot across all metrics, while QuaRot+ (with dimension expansion) enables compatibility with Qwen models.

Model	**Method**	**PPL↓**	**PIQA**	**HS**	**ARC-E**	**ARC-C**	**Wino**	**Lambada**	**AVG↑**
Mistral 7B Inst v0.3	FP16	5.49	71.27	74.6	67.94	74.27	58.96	82.66	71.62
	QuaRot	5.98	67.7	71.45	63.94	69.53	55.2	79.46	67.88
	QuaRot + GBS	**5.75**	**70.55**	**74.44**	**66.66**	**71.82**	**56.66**	**80.77**	**70.15**
Mistral 7B v0.1	FP16	5.25	72.49	75.59	69.4	73.95	54.86	80.18	71.08
	QuaRot	5.82	67.62	71.88	63.36	70.01	48.98	76.6	66.41
	QuaRot + GBS	**5.57**	**71.47**	**74.95**	**68**	**72.22**	**51.54**	**79.21**	**69.56**
Llama2 7B	FP16	5.47	71.08	73.9	68.25	68.98	46.33	74.58	67.19
	QuaRot	6.21	64.65	69.09	60.22	64.64	**43.17**	69.78	61.92
	QuaRot + GBS	**6.04**	**65.64**	**69.88**	**61.4**	**66.46**	42.32	**70.75**	**62.74**
Llama3 8B	FP16	6.13	72.62	76.01	69.22	72.93	53.41	77.69	70.3
	QuaRot	8.33	61.66	66.27	57.05	64.72	42.06	68.06	59.97
	QuaRot + GBS	**7.4**	**67.87**	**72.02**	**63.73**	**71.03**	**45.9**	**73.7**	**65.71**
Qwen2.5 7B Inst	FP16	7.45	66.23	69.73	62.74	70.56	55.12	81.02	67.57
	QuaRot	-	-	-	-	-	-	-	-
	QuaRot$^+$	9.21	56.66	59.31	54.01	63.54	48.89	69.78	58.7
	QuaRot$^+$ + GBS	**8.23**	**62.58**	**64.91**	**60.24**	**66.61**	**49.4**	**72.39**	**62.69**
Qwen2.5 1.5B Inst	FP16	9.64	58.09	61.21	54.98	63.3	46.59	75.8	60.0
	QuaRot	14.44	39.05	40.23	37.86	54.85	35.75	58.71	44.41
	QuaRot + GBS	**12.05**	**43.94**	**45.24**	**42.64**	**58.64**	**39.33**	**65.61**	**49.23**

Table 2: Results in 3-bit WAKV quantization on seven benchmarks (perplexity, PIQA, HellaSwag, ARC-Easy, ARC-Challenge, Winogrande, and LAMBADA) and report average success rates. Our GBS method outperforms QuaRot across all metrics, while QuaRot+ (with dimension expansion) enables compatibility with Qwen models.

Model	**Method**	**PPL↓**	**PIQA**	**HS**	**ARC-E**	**ARC-C**	**Wino**	**Lambada**	**AVG↑**
Mistral 7B Inst v0.3	FP16	5.49	71.27	74.6	67.94	74.27	58.96	82.66	71.62
	QuaRot	38.28	7.66	9.49	5.82	51.46	23.55	34.39	22.06
	QuaRot + GBS	**7.04**	**62.17**	**66.93**	**57.4**	**61.56**	**46.42**	**73.44**	**61.32**
Mistral 7B v0.1	FP16	5.25	72.49	75.59	69.4	73.95	54.86	80.18	71.08
	QuaRot	100.85	3.07	4.13	2.0	48.62	22.18	30.6	18.43
	QuaRot + GBS	**7.31**	**59.22**	**64.41**	**54.03**	**63.3**	**40.53**	**68.39**	**58.31**
Llama2 7B	FP16	5.47	71.08	73.9	68.25	68.98	46.33	74.58	67.19
	QuaRot	332.56	0.25	0.47	0.04	51.14	26.11	30.39	18.07
	QuaRot + GBS	**9.18**	**39.03**	**49.91**	**28.14**	**56.12**	**31.91**	**54.92**	**43.34**
Llama3 8B	FP16	6.13	72.62	76.01	69.22	72.93	53.41	77.69	70.3
	QuaRot	1315	0.05	0.08	0.02	49.41	23.72	27.78	16.84
	QuaRot + GBS	**12.62**	**44.92**	**50.32**	**39.51**	**60.22**	**32.85**	**53.7**	**46.92**
Qwen2.5 7B Inst	FP16	7.45	66.23	69.73	62.74	70.56	55.12	81.02	67.57
	QuaRot	-	-	-	-	-	-	-	-
	QuaRot$^+$	251	1.14	1.14	1.14	49.33	25.0	32.15	18.32
	QuaRot$^+$ + GBS	**12.33**	**41.9**	**42.62**	**41.18**	**56.59**	**40.53**	**61.83**	**47.44**
Qwen2.5 1.5B Inst	FP16	9.64	58.09	61.21	54.98	63.3	46.59	75.8	60.0
	QuaRot	3411	0.06	0.12	0.0	49.72	**23.98**	27.99	16.98
	QuaRot + GBS	**34.97**	**13.84**	**14.81**	**12.87**	**52.17**	23.72	**38.89**	**26.05**

We also evaluated GBS using two additional methods: SpinQuant (Liu et al. (2024)) and DFRot (Xiang & Zhang (2024)), both relying on rotation matrices (see Appendix F). Our binary search approach, particularly when restricted to 3 bits, significantly enhances their performance, thus validating the broad effectiveness of our method

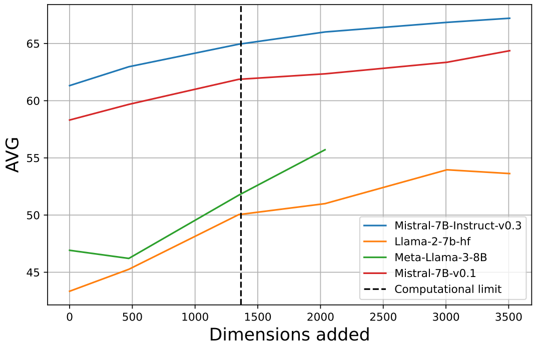


Figure 2: Effect of expanding dimensions on 6 benchmarks average (AVG) for different models in 3 bits WAKV quantization and the computational limit of Lemma 4.1. Due to memory constraints on GPU A100 we could not increase more than 2036 dimensions for LLaMA3-8B.

5.2.2 Matrix expansion effect

We now study the impact of expanding dimensions on performance. Figure 2 show the evolution of AVG with the number of dimensions added to our tokens and we clearly see the positive impact on performances. We can reach 68.95% of accuracy for Mistral-7B Instruct but at a very high computational cost.

Another beneficial aspect of dimension expansion is seen in Group Local Rotation, introduced in QuaRot and explored in LightRot (Kim et al. (2025)). This technique involves decomposing a tensor into smaller sub-tensors and applying the same small power-of-2 Hadamard matrix to each of these sub-tensors. This approach leverages efficient Hadamard transforms (as introduced in Section 2.3.2) and significantly speeds up inference. Particularly for MLP layers that often operate in high-dimensional spaces, expanding dimensions can help identify a more suitable divisor, resulting in efficient power-of-2 sub-tensors.

6 Conclusion

In this work, we introduced an approach to optimize the quantization of LLMs using Gradual Binary Search and Hadamard matrices. Our method achieves efficient 3-bit quantization for weights, activations, and key-value caches, significantly improving model performance. We also theoretically demonstrated that Hadamard matrices are more effective than random rotation matrices in reducing extreme values in activations.

We also extended the use of rotation matrices to support non-power-of-2 embedding dimensions using the Paley algorithm and dimension expansion. This generalization allows our method to be applied to various architectures, including those with unique embedding dimensions. Experimental results on models from the Mistral library, LLaMA architecture, and Qwen show the effectiveness of our approach, outperforming existing methods.

Overall, our findings suggest that GBS and Hadamard matrices have great potential for advancing LLM quantization, making them more suitable for resource-constrained devices. Future work will explore mix computation and combining GBS with other methods.

7 Limitations

As explained in the previous part expanding dimensions has a big computational cost and it worsen with context length that is why we need to be aware of the expanding limit. One potential solution is to implement a Mixed Computation pipeline, where dimensions are only expanded in specific layers based on the presence of outliers, thereby substantially reducing computational overhead.

Another challenge arises with GBS, which involves computing perplexity at each step, a process that can be time-consuming for our method, sometimes taking several days. To mitigate this, we tried different dataset sizes and maximum binary search iterations to reduce the computation time which is necessary to scale GBS to bigger models.

References

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress Large Language Models by Deleting Rows and Columns, February 2024a. URL `http://arxiv.org/abs/2401.15024`. arXiv:2401.15024.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs, October 2024b. URL `http://arxiv.org/abs/2404.00456`. arXiv:2404.00456 [cs].

Jerry Chee, Volodymyr Kuleshov, and Yaohui Cai. QuIP: 2-Bit Quantization of Large Language Models With Guarantees.

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. PrefixQuant: Static Quantization Beats Dynamic through Prefixed Outliers in LLMs, October 2024. URL `http://arxiv.org/abs/2410.05265`. arXiv:2410.05265.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I.-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks, July 2018. URL `http://arxiv.org/abs/1805.06085`. arXiv:1805.06085 [cs].

Laurens De Haan and Ana Ferreira. *Extreme Value Theory*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2006. ISBN 978-0-387-23946-0 978-0-387-34471-3. doi: 10.1007/0-387-34471-3. URL `http://link.springer.com/10.1007/0-387-34471-3`.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale.

Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable Model Compression via Pseudo Quantization Noise, October 2022. URL `http://arxiv.org/abs/2104.09987`. arXiv:2104.09987 [cs, stat].

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned Step Size Quantization, May 2020. URL `http://arxiv.org/abs/1902.08153`. arXiv:1902.08153 [cs, stat].

Giuseppe Franco, Pablo Monteagudo-Lago, Ian Colbert, Nicholas Fraser, and Michaela Blott. Improving Quantization with Post-Training Model Expansion, March 2025. URL `http://arxiv.org/abs/2503.17513`. arXiv:2503.17513 [cs].

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers, March 2023. URL `http://arxiv.org/abs/2210.17323`. arXiv:2210.17323 [cs].

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference, June 2021. URL `http://arxiv.org/abs/2103.13630`. arXiv:2103.13630 [cs].

Yunhui Guo. A Survey on Methods and Theories of Quantized Neural Networks, December 2018. URL `http://arxiv.org/abs/1808.04752`. arXiv:1808.04752 [cs, stat].

K. J. Horadam. *Hadamard Matrices and Their Applications*. Princeton University Press, January 2012. ISBN 978-1-4008-4290-2. Google-Books-ID: oR_HDgAAQBAJ.

Xijie Huang, Zechun Liu, Shih-Yang Liu, and Kwang-Ting Cheng. RoLoRA: Fine-tuning Rotated Outlier-free LLMs for Effective Weight-Activation Quantization, July 2024. URL `http://arxiv.org/abs/2407.08044`. arXiv:2407.08044 [cs].

Sangjin Kim, Yuseon Choi, Jungjun Oh, Byeongcheol Kim, and Hoi-Jun Yoo. LightRot: A Light-weighted Rotation Scheme and Architecture for Accurate Low-bit Large Language Model Inference. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 1–1, 2025. ISSN 2156-3365. doi: 10.1109/JETCAS.2025.3558300. URL `https://ieeexplore.ieee.org/document/10950449/`.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing Outliers via Dual Transformation Makes Stronger Quantized LLMs.

Ji Lin, Chuang Gan, and Song Han. Defensive Quantization: When Efficiency Meets Robustness, April 2019. URL http://arxiv.org/abs/1904.08444. arXiv:1904.08444 [cs, stat].

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations, May 2024. URL http://arxiv.org/abs/2405.16406. arXiv:2405.16406 [cs].

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits, February 2024. URL http://arxiv.org/abs/2402.17764. arXiv:2402.17764 [cs].

Lucas Maisonnave, Cyril Moineau, Olivier Bichler, and Fabrice Rastello. Applying maximum entropy principle on quantized neural networks correlates with high accuracy.

Lucas Maisonnave, Cyril Moineau, Olivier Bichler, and Fabrice Rastello. Precision Where It Matters: A Novel Spike Aware Mixed-Precision Quantization Strategy for LLaMA-based Language Models, April 2025. URL http://arxiv.org/abs/2504.21553. arXiv:2504.21553 [cs].

Aniruddha Nrusimha, Mayank Mishra, Naigang Wang, Dan Alistarh, Rameswar Panda, and Yoon Kim. Mitigating the Impact of Outlier Channels for Language Model Quantization with Activation Regularization, April 2024. URL http://arxiv.org/abs/2404.03605. arXiv:2404.03605 [cs] version: 1.

Seungwoo Son, Wonpyo Park, Woohyun Han, Kyuyeun Kim, and Jaeho Lee. Prefixing Attention Sinks can Mitigate Activation Outliers for Large Language Model Quantization. June 2024. URL https://www.semanticscholar.org/paper/1601ad7616681ed9d7e1b9a04b64c1ad9c7196c7.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 2018. ISBN 978-1-108-41519-4. doi: 10.1017/9781108231596. URL https://www.cambridge.org/core/books/highdimensional-probability/797C466DA29743D2C8213493BD2D2102.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit Transformers for Large Language Models, October 2023. URL http://arxiv.org/abs/2310.11453. arXiv:2310.11453 [cs].

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier Suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling, October 2023. URL http://arxiv.org/abs/2304.09145. arXiv:2304.09145 [cs].

Jingyang Xiang and Sai Qian Zhang. DFRot: Achieving Outlier-Free and Massive Activation-Free for Rotated LLMs with Refined Rotation, December 2024. URL http://arxiv.org/abs/2412.00648. arXiv:2412.00648 [cs].

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 38087–38099. PMLR, July 2023. URL https://proceedings.mlr.press/v202/xiao23c.html. ISSN: 2640-3498.

Dawei Yang, Ning He, Xing Hu, Zhihang Yuan, Jiangyong Yu, Chen Xu, and Zhe Jiang. Post-Training Quantization for Re-parameterization via Coarse & Fine Weight Splitting, December 2023. URL http://arxiv.org/abs/2312.10588. arXiv:2312.10588 [cs].

Linjie Yang and Qing Jin. FracBits: Mixed Precision Quantization via Fractional Bit-Widths. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10612–10620, May 2021. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v35i12.17269. URL https://ojs.aaai.org/index.php/AAAI/article/view/17269.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained Ternary Quantization, February 2017. URL http://arxiv.org/abs/1612.01064. arXiv:1612.01064 [cs].

A Theoretical proofs

A.1 Hadamard matrices

A.1.1 Proof

Proof of Lemma 3.1. We define a Hadamard matrix as a rotation matrix with values equal to 1 or -1 only. By definition, the equality $HH^T = I_n$ must be respected, but without normalization, we have $HH^T = nI_n$. If we decide to normalize this Hadamard matrix by a factor of $\frac{1}{\sqrt{n}}$, we obtain the identity by multiplying it by its transpose. In the case of a vector $x = \epsilon + \sum_{j=1}^{k} ce_{p_j}$ with $c >> ||\epsilon||$, we have

$$Hx = H\epsilon + H\sum_{j=1}^{k} ce_{p_j} = H\epsilon + \sum_{j=1}^{k} cH_{:,p_j}$$

We consider $H\epsilon$ negligible in high dimension compare to the rest and for a normalized Hadamard matrix, each element is $\pm\frac{1}{\sqrt{n}}$. applying a Hadamard matrix to it amounts to multiplying the maximum absolute value by $\frac{1}{\sqrt{n}}$ since all the values of H are either $\frac{1}{\sqrt{n}}$ or $-\frac{1}{\sqrt{n}}$. The worst-case scenario (maximum amplitude) occurs when all signs of the elements H_{i,p_j} are identical for some row i. The first row of a Hadamard matrix is always composed of positive values, therefore:

$$\mu(Hx) \sim \sum_{j=1}^{k} \frac{c}{\sqrt{n}} \sim \frac{kc}{\sqrt{n}}$$

□

Proof of Lemma 3.2. Let Q be a rotation matrix drawn on the unit sphere $\mathcal{S}^{n-1}$. We assume that the problem is in high dimension, which allows us to approximate the distribution of the elements of the matrix Q:

$$Q_{ij} \sim \mathcal{N}\left(0, \frac{1}{n}\right)$$

This theorem is a classic result of high-dimensional probability theory (Vershynin (2018)). We can use the same decomposition of x to have

$$Qx = Q\epsilon + \sum_{j=1}^{k} cQ_{:,p_j}$$

This time- for each row i, Q_{i,p_j} is a random variable so we can't just add the contribution of each maximum. We define $S_i = \sum_{i,j} Q_{ij}$ By the *variance addition property* for independent random variables, we have $\text{Var}\left(\sum_{j=1}^{k} X_j\right) = \sum_{j=1}^{k} \text{Var}(X_j)$, leading to

$$S_i \sim \mathcal{N}\left(0, \frac{k}{n}\right)$$

From this approximation we can use the fundamental properties of the extreme values of a normal law. Indeed, for all $i \leq n$, $Z_i \sim \mathcal{N}(0, 1)$, then $S_i = Z_i\sqrt{\frac{k}{n}}$. We can show (De Haan & Ferreira (2006)) that

$$\mathbb{E}\left[\max_{1\leq i\leq n} |Z_i|\right] = \sqrt{2\log n} \tag{3}$$

Therefore,

$$\mathbb{E}\left[\max_{1\leq i\leq n} |S_i|\right] = \sqrt{\frac{2k\log n}{n}} \tag{4}$$

Using Talagrand's inequality for a Lipschitz function (Qx is indeed a Lipschitz function) we have:

$$\mathbb{P}\left(\left|\max_{1\leq i\leq n} |S_i| - \sqrt{\frac{2k\log n}{n}}\right| > \epsilon\right) \leq 2e^{-Cn\epsilon^2} \tag{5}$$

With $C > 0$. Thus, for sufficiently large n, we have a very high probability of having:

$$\max_{1\leq i\leq n} |S_i| = \sqrt{\frac{2k\log n}{n}} \tag{6}$$

We now use this result when applying Q to a vector x with k outliers

$$\mu(Qx) = \max_{1\leq i\leq n} |(Qx)_i| \sim c \max_{1\leq i\leq n} |\sum_{j=1}^{k} Q_{i,p_j}| \sim c\sqrt{\frac{2k\log n}{n}}$$

Finally, using Lemmas 3.1 and 3.2, we show that for sufficiently large n

$$\frac{\mu(Hx)}{\mu(Qx)} \sim \sqrt{\frac{k}{2\log n}}$$

□

A.1.2 Experimental Verifications

We conduct a comprehensive empirical study (Figure 3) to compare the outlier reduction capabilities of Hadamard matrices versus random orthogonal rotation matrices across varying dimensions and sparsity levels. Our experimental framework generates synthetic signals of dimension $n \in \{128, 256, 512, 1024, 2048\}$ containing $k \in \{1, 2, 4, 8\}$ randomly positioned outliers with amplitude 1000.0, embedded within Gaussian noise ($\sigma = 0.1$). For each configuration, we perform 50 independent trials to ensure statistical reliability. The Hadamard matrices are constructed using the standard recursive definition for power-of-two dimensions, normalized by $\frac{1}{\sqrt{n}}$, while random orthogonal matrices are drawn from the Haar measure using Scipy's ortho_group distribution. For each trial, we compute the reduction factor as the ratio between the maximum absolute value of the transformed signal and the original signal, providing a direct measure of outlier suppression effectiveness. We compare experimental results against theoretical predictions: $\frac{k}{\sqrt{n}}$ for Hadamard matrices and $\alpha\sqrt{\frac{2k\log n}{n}}$ for rotation matrices (with empirically determined correction factor $\alpha = 0.9$).

It clearly shows that the theoretical and experimental curves follow each other perfectly, which seems to confirm the previously demonstrated theorems. Hadamard matrices are therefore theoretically and experimentally the most suitable matrices for reducing the impact of many outliers in a vector. But we can see for $k = 8$ the gap between Rotations and Hadamard is already smaller which suggests in the case of many outliers like 100 or more Hadamard matrices could be less efficient.

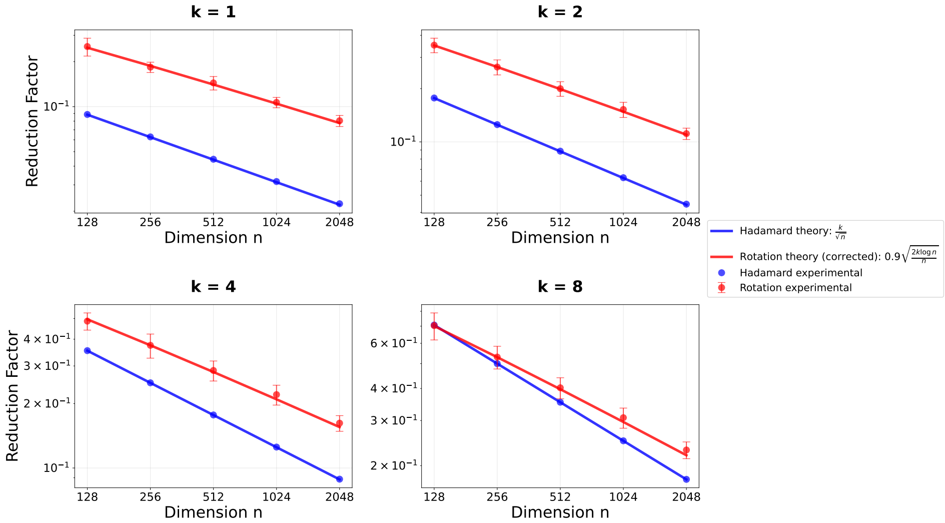


Figure 3: Outlier reduction factor comparison between Hadamard and rotation matrices. Results for $k \in \{1, 2, 4, 8\}$ outliers across dimensions $n \in \{128, 256, 512, 1024, 2048\}$. Solid lines show theoretical predictions: blue for Hadamard ($k/\sqrt{n}$), red for rotation matrices ($0.9\sqrt{2k\log n/n}$). Circles represent experimental data (50 trials, error bars = $\pm 1\sigma$). Hadamard matrices demonstrate superior outlier reduction.

A.2 Increasing dimensions

Proof of Lemma 4.1. We define BitOps as the function that compute the number of operations for a matrix multiplication: $\text{BitOps}(AB) = mn^2pb^2$. And we want to find a condition that ensure

$$\text{BitOps}(A'B') \leq \text{BitOps}(AB) \tag{7}$$

$$\Rightarrow \quad m(n+d)^2pb'^2 \leq mn^2pb^2 \tag{8}$$

$$\Rightarrow \quad (n+d)^2b'^2 \leq n^2b^2 \tag{9}$$

$$\Rightarrow \quad (n+d)b' \leq nb \tag{10}$$

$$\Rightarrow \quad nb' + db' \leq nb \tag{11}$$

$$\Rightarrow \quad d \leq \frac{n(b-b')}{b'} \tag{12}$$

□

B Paley Algorithm

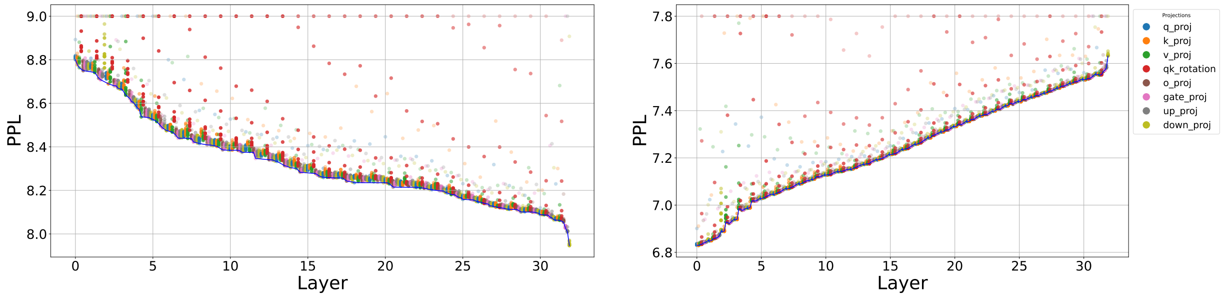
C Gradual Binary Search process

In this analysis, we examine the evolution of clipping ratios through GBS to better understand the dynamics of these parameters in LLMs. Figure 4 illustrates the perplexity (PPL) evolution during the optimization of a LLaMA3-8B model quantized to 4 bits. The graph displays the various tested values for each projection, optimized under two different configurations: starting the model in FP16 (blue line) and initiating the process in 4 bits. It is clear that starting in FP16 yields a better PPL on the training set of WikiText2, achieving 7.62, compared to starting in 4 bits, which results in a PPL of 7.94. On the test set we have the same dynamic with a PPL of 7.4 starting in FP16 and 7.69 starting in 4 bits.

Figure 5 illustrates the final configuration achieved by GBS for the same architecture, starting the process in both FP16 and 4-bit precision. It is clear that initiating in 4 bits results in a significantly more unstable

Algorithm 2 Hadamard Matrix Construction using the Paley Method

```
Require: A prime number $p$
Ensure: A Hadamard matrix $H$ of order $p+1$
$n \leftarrow p+1$                                                ▷ Determine the order of the matrix
Initialize $H$ as a $n \times n$ matrix with all entries set to 1        ▷ Start with a matrix of all ones
for $i \leftarrow 1$ to $n-1$ do                                  ▷ Loop over rows (except the first row)
    $H[i,0] \leftarrow -1$                                        ▷ Set the first column entry to -1 for current row
    $H[0,i] \leftarrow -1$                                        ▷ Set the first row entry to -1 for current column
    for $j \leftarrow 1$ to $n-1$ do                              ▷ Loop over columns (except the first column)
        if $i = j$ then
            $H[i,j] \leftarrow -1$                                ▷ Set diagonal entries to -1
        else
            $H[i,j] \leftarrow$ legendre_symbol($(i-1)-(j-1), p$)
        end if
    end for
end for
return $H$                                                        ▷ Return the constructed Hadamard matrix
```



(a) GBS starting in 4 bits

(b) GBS starting in FP16

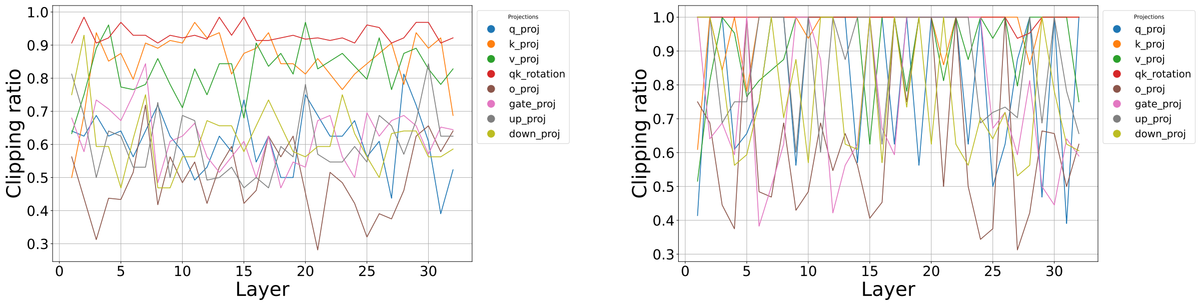
Figure 4: PPL vs Layer during Gradual Binary Search on 10% of Train WikiText2 for a LLaMA3-8B in 4-bit quantization and rotated with QuaRot. For better visualization we set a maximum PPL to 9. Points opacity represents the clipping ratio, the value is closer to 0 as transparency increases

configuration compared to starting in FP16. Many values remain at 1, and there is a high variance, indicating that the algorithm struggles to find a stable configuration when the entire model is in 4 bits. It also appears to have difficulty understanding the impact of small changes in the clipping ratio.

In contrast, starting in FP16 results in a stable configuration for every projection, with distinct dynamics. For instance, the qk_rotation projection exhibits minimal changes in the clipping ratio, with most layers close to 1. Conversely, the o_proj projection has values below 0.3, suggesting that clipping to 30% of the maximum value can enhance performance. This figure underscores the importance of GBS in improving the model's quality by identifying the optimal clipping configuration, which is clearly not only ones.

D Perplexity as objective

Perplexity is the central part of our optimization, it drives our search and it is supposed to reach a configuration which will performs better than all others with a bigger PPL. In figure 6 we can see how the average value on 6 benchmarks evolves with the perplexity. It clearly appears that a smaller PPL usually represents a better AVG.



(a) GBS starting in FP16

(b) GBS starting in 4 bits

Figure 5: Final configurations obtained with GBS started in 4 bits and in FP16 for a LLaMA3-8B

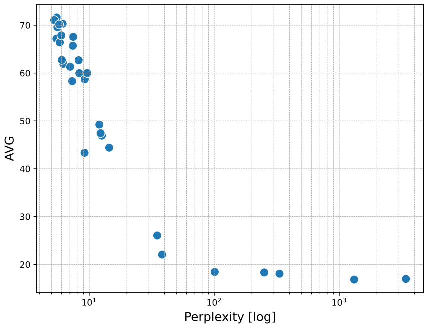


Figure 6: AVG over Perplexity for all results obtained in Tab 2 and 1

E Computation time

In table 3 we can clearly see we can improve the computation by changing 2 parameters : the size of the dataset and max_iter the maximum number iterations allowed for the bianry search. On a Llama2-7B we can reduce to 10h without a significant drop of perplexity. In 3 bits the model is more sensitive and we can see a drop but still remaining well better than previous state of the art methods. We can also drastically reduce the computation time for Llama3-8B with a small PPL increase (+0.13 in best case in 4 bits). While the results aren't yet optimal, they clearly demonstrate that more effective clipping ratio optimization is crucial for substantial performance gains.

Table 3: Ablation study on α (% of the train set of WikiText2) and max_iter the maximum number of iteration for the binary search.

Model	Bit	α (%)	max_iter	time(h)	PPL
Llama2 7B	4	10	10	50	6.04
			8	31.9	6.07
			7	28.6	6.08
			6	24.7	6.08
			5	21.02	6.08
			4	17.42	6.09
			3	14.07	6.37
		5	5	10.89	6.11
	3	10	10	50	9.18
			5	19.5	10.39
		5	5	10	10.58
Llama3 8B	4	10	10	80	7.4
			5	21	7.53
		5	5	10	7.56
	3	10	10	80	12.62
			5	19.44	15.65
		5	5	10	15.72

F More results

F.1 SpinQuant

Table 4: Results in 4 bits WAKV quantization on perplexity (PPL), PIQA, hellaswag (HS), arc-easy (ARC-E), arc-challenge (ARC-C), winogrande (WINO) and lambada, we also compute the average value (AVG) which represents a % of success. We compare our method, GBS, with SpinQuant and clearly observe that GBS outperforms SpinQuant across almost all computed metrics.

Model	Method	PPL↓	PIQA	HS	ARC-E	ARC-C	Wino	Lambada	AVG↑
Mistral 7B Inst v0.3	FP16	5.49	71.27	74.6	67.94	74.27	58.96	82.66	71.62
	SpinQuant	5.88	69	72.66	65.34	69.93	55.12	78.41	68.41
	SpinQuant + GBS	**5.83**	**69.01**	**72.66**	**65.36**	**72.14**	**56.91**	**79.5**	**69.26**
Mistral 7B v0.1	FP16	5.25	72.49	75.59	69.4	73.95	54.86	80.18	71.08
	SpinQuant	5.71	69.55	**73.45**	65.65	69.38	**48.98**	**76.98**	67.33
	SpinQuant + GBS	**5.62**	**70.02**	73.37	**66.66**	**71.19**	48.29	76.47	**67.67**
Llama2 7B	FP16	5.47	71.08	73.9	68.25	68.98	46.33	74.58	67.19
	SpinQuant	6.57	61.73	68.46	55.0	63.46	40.61	67.63	59.48
	SpinQuant + GBS	**6.15**	**63.24**	**69.01**	**57.46**	**65.04**	**41.04**	**68.43**	**60.7**
Llama3 8B	FP16	6.13	72.62	76.01	69.22	72.93	53.41	77.69	70.3
	SpinQuant	7.97	**65.11**	**69.01**	**61.21**	66.61	45.22	72.52	63.28
	SpinQuant + GBS	**7.69**	63.84	67.84	59.83	**70.4**	**47.35**	**73.36**	**63.77**

Table 5: Results in 3 bits WAKV quantization on perplexity (PPL), PIQA, hellaswag (HS), arc-easy (ARC-E), arc-challenge (ARC-C), winogrande (WINO) and lambada, we also compute the average value (AVG) which represents a % of success. We compare our method, GBS, with SpinQuant and clearly observe that GBS outperforms SpinQuant across almost all computed metrics.

Model	Method	PPL↓	PIQA	HS	ARC-E	ARC-C	Wino	Lambada	AVG↑
Mistral 7B Inst v0.3	FP16	5.49	71.27	74.6	67.94	74.27	58.96	82.66	71.62
	SpinQuant	14.31	20.39	28.39	12.38	49.72	24.57	40.57	29.34
	SpinQuant + GBS	**8.11**	**52.93**	**58.99**	**46.87**	**60.06**	**40.78**	**66.67**	**54.38**
Mistral 7B v0.1	FP16	5.25	72.49	75.59	69.4	73.95	54.86	80.18	71.08
	SpinQuant	18.88	16.58	21.97	11.2	51.3	24.23	37.67	27.16
	SpinQuant + GBS	**7.98**	**52.81**	**60.59**	**45.04**	**59.27**	**35.67**	**63.68**	**52.84**
Llama2 7B	FP16	5.47	71.08	73.9	68.25	68.98	46.33	74.58	67.19
	SpinQuant	425.2	0.24	0.45	0.04	**51.46**	28.33	27.57	18.02
	SpinQuant + GBS	**15.65**	**20.77**	**26.51**	**15.04**	50.67	**26.19**	**42.93**	**30.35**
Llama3 8B	FP16	6.13	72.62	76.01	69.22	72.93	53.41	77.69	70.3
	SpinQuant	316.6	2.71	3.1	2.31	50.51	22.35	29.0	18.33
	SpinQuant + GBS	**20.26**	**21.42**	**23.95**	**18.9**	**54.38**	**26.96**	**43.1**	**31.45**

F.2 DFRot

Table 6: Results in 4 bits WAKV quantization on perplexity (PPL), PIQA, hellaswag (HS), arc-easy (ARC-E), arc-challenge (ARC-C), winogrande (WINO) and lambada, we also compute the average value (AVG) which represents a % of success. We compare our method, GBS, with DFRot and clearly observe that GBS outperforms DFRot across all computed metrics.

Model	Method	PPL↓	PIQA	HS	ARC-E	ARC-C	Wino	Lambada	AVG↑
Mistral 7B Inst v0.3	FP16	5.49	71.27	74.6	67.94	74.27	58.96	82.66	71.62
	DFRot	5.94	68.11	79.43	68.5	72.42	64.58	**80.3**	72.22
	DFRot + GBS	**5.81**	**71.35**	**81.23**	**69.3**	**73.18**	**65.42**	80.13	**73.43**
Mistral 7B v0.1	FP16	5.25	72.49	75.59	69.4	73.95	54.86	80.18	71.08
	DFRot	5.75	**71.03**	78.94	69.83	74.03	65.63	78.28	72.96
	DFRot + GBS	**5.62**	70.88	**80.9**	**70.93**	**75.0**	**66.85**	**78.85**	**73.9**
Llama2 7B	FP16	5.47	71.08	73.9	68.25	68.98	46.33	74.58	67.19
	DFRot	6.23	65.04	76.66	65.75	69.47	62.02	72.61	68.59
	DFRot + GBS	**6.05**	**65.67**	**77.75**	**66.41**	**69.63**	**63.19**	**72.74**	**69.23**
Llama3 8B	FP16	6.13	72.62	76.01	69.22	72.93	53.41	77.69	70.3
	DFRot	7.95	68.11	76.01	64.92	68.5	61.34	74.17	68.84
	DFRot + GBS	**7.56**	**72.53**	**76.82**	**66.35**	**69.53**	**63.17**	**75.0**	**70.57**

Table 7: Results in 3 bits WAKV quantization on perplexity (PPL), PIQA, hellaswag (HS), arc-easy (ARC-E), arc-challenge (ARC-C), winogrande (WINO) and lambada, we also compute the average value (AVG) which represents a % of success. We compare our method, GBS, with DFRot and clearly observe that GBS outperforms DFRot across all computed metrics.

Model	**Method**	**PPL↓**	**PIQA**	**HS**	**ARC-E**	**ARC-C**	**Wino**	**Lambada**	**AVG↑**
Mistral 7B Inst v0.3	FP16	5.49	71.27	74.6	67.94	74.27	58.96	82.66	71.62
	DFRot	11.26	53.28	67.57	35.6	40.33	30.88	57.57	47.54
	DFRot + GBS	**7.58**	**63.22**	**75.35**	**59.32**	**65.19**	**53.46**	**71.73**	**64.71**
Mistral 7B v0.1	FP16	5.25	72.49	75.59	69.4	73.95	54.86	80.18	71.08
	DFRot	13.63	55.01	65.45	27.89	32.52	23.25	50.63	42.46
	DFRot + GBS	**7.64**	**62.83**	**73.72**	**56.99**	**64.33**	**49.64**	**68.12**	**62.6**
Llama2 7B	FP16	5.47	71.08	73.9	68.25	68.98	46.33	74.58	67.19
	DFRot	26.64	49.96	58.81	13.19	14.81	11.57	39.14	31.25
	DFRot + GBS	**10.96**	**56.12**	**66.81**	**34.43**	**40.4**	**28.45**	**55.01**	**46.87**
Llama3 8B	FP16	6.13	72.62	76.01	69.22	72.93	53.41	77.69	70.3
	DFRot	140.78	52.41	54.62	2.82	3.38	2.27	31.85	24.56
	DFRot + GBS	**22.14**	**54.85**	**61.92**	**25.53**	**29.58**	**21.48**	**48.67**	**40.34**