FBQuant: FeedBack Quantization for Large Language Models

Yijiang Liu[1] , **Hengyu Fang**[1] , **Liulu He**[1] , **Rongyu Zhang**[1] , **Yichuan Bai**[1] ,
Yuan Du[1,2] and **Li Du**[1,2,✉]
[1]School of Electronic Science and Engineering, Nanjing University
[2]Interdisciplinary Research Center for Future Intelligent Chips (Chip-X), Nanjing University, Suzhou
{liuyijiang, hengyufang, heliulu, rongyuzhang, baiyichuan}@smail.nju.edu.cn,
{yuandu, ldu}@nju.edu.cn

Abstract

Deploying Large Language Models (LLMs) on edge devices is increasingly important, as it eliminates reliance on network connections, reduces expensive API calls, and enhances user privacy. However, on-device deployment is challenging due to the limited computational resources of edge devices. In particular, the key bottleneck stems from memory bandwidth constraints related to weight loading. Weight-only quantization effectively reduces memory access, yet often induces significant accuracy degradation. Recent efforts to incorporate sub-branches have shown promise for mitigating quantization errors, but these methods either lack robust optimization strategies or rely on suboptimal objectives. To address these gaps, we propose FeedBack Quantization (FBQuant), a novel approach inspired by negative feedback mechanisms in automatic control. FBQuant inherently ensures that the reconstructed weights remain bounded by the quantization process, thereby reducing the risk of overfitting. To further offset the additional latency introduced by sub-branches, we develop an efficient CUDA kernel that decreases 60% of extra inference time. Comprehensive experiments demonstrate the efficiency and effectiveness of FBQuant across various LLMs. Notably, for 3-bit Llama2-7B, FBQuant improves zero-shot accuracy by 1.2%.

1 Introduction

Large Language Models (LLMs) [Touvron *et al.*, 2023] have demonstrated remarkable capabilities in natural language processing, driving significant advancements in the field. while their extensive parameter counts enable this high-level performance, they also pose substantial deployment challenges related to memory access, storage, and computation. These issues challenges even more critical for personal, localized applications, where privacy concerns often preclude the use of cloud-end LLM APIs, and where most personal devices lack high-performance accelerators (e.g., top-tier GPUs). Furthermore, as AI systems increasingly rely on LLM-based agents [Talebirad and Nadiri, 2023; Li *et al.*, 2023b] to handle ever more complex tasks, the computational demands continue to intensify, raising unprecedented barriers to on-device deployment.

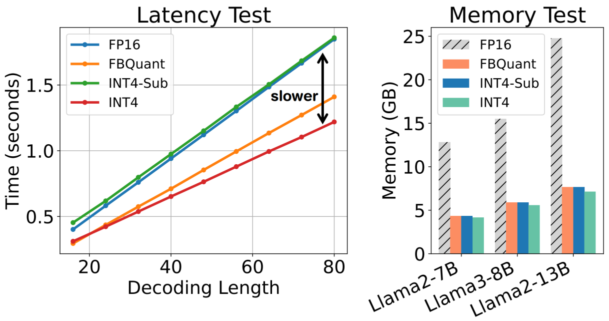

Figure 1: Impact of weight-only quantization on the RTX 3090 GPU. (Left) For Llama2-7B, the INT4 model processes 1,024 tokens for prefilling and 80 new tokens for decoding in only 60% of the time required by FP16. (Right) After loading to the GPU device, the INT4 model consumes just 25% of the memory used by FP16.

One defining characteristic of the on-device LLMs is the consistently small batch size (in most cases, a batch size of one) [Lin *et al.*, 2024b]. Consequently, inference is predominantly a memory-bandwidth-bound operation constrained by weight loading. Under these conditions, weight-only quantization emerges as a highly effective optimization strategy (as shown in Fig. 1). By storing weights in INT3/4 while retaining inputs and outputs in floating-point, memory requirements and bandwidth overhead are substantially reduced. Moreover, this technique can be broadly applied to various hardware platforms, such as desktops, laptops, and mobile phones, offering a cost-effective solution that balances performance and affordability.

Existing optimization methods of weight-only quantization for LLMs can be broadly classified into three categories, as illustrated in Fig. 2, **Clamping, Rotation, and Sub-branching**: (a) Clamping methods [Lin *et al.*, 2024b; Shao *et al.*, 2023] refine the quantization scale by restricting the value range, improving the representation of smaller weights while sacrificing outliers. (b) Rotation-based ap-

✉ denotes the corresponding author.

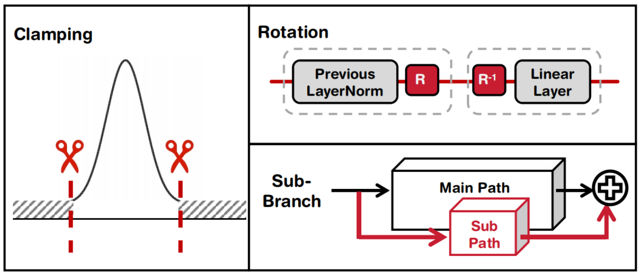

Figure 2: Three categories of optimization methods for weight-only quantization: Clamping, Rotation, and Sub-branching.

proaches [Liu *et al.*, 2024b; Lin *et al.*, 2024a] apply rotation-equivalent transformations to shift quantization challenges from weights (which are quantized) to activations (which remain in high precision). (c) Sub-branching [Li *et al.*, 2023a; Li *et al.*, 2024] methods introduce a parallel branch alongside the quantized main path to compensate for quantization errors. Despite extensive research into clamping and rotation techniques, the quantization-induced accuracy degradation remains a critical concern. Recently, sub-branching methods have gained attention as an orthogonal solution for mitigating quantization errors. The key idea is to split each quantized layer into two parallel paths: a main path that retains the quantized weights and a sub-branch that compensates for quantization losses. Recent approaches, such as CALDERA [Saha *et al.*, 2024] and EoRA[Liu *et al.*, 2024a], construct sub-branches using the LoRA [Hu *et al.*, 2021] framework, but these methods can overfit to limited calibration data or rely on ill-posed optimization objectives (see Sec. 3). Another significant challenge of sub-branching methods is the unexpected increase in inference latency, despite the sub-branches only introducing a small number of operations. This delay is primarily driven by memory access bottlenecks, as the sub-branch frequently reads and writes input activations, intermediate results, and layer outputs.

In this work, we propose a novel method called **F**eed**B**ack **Quant**ization (**FBQuant**), tackling both the optimization problem and the latency overhead introduced by sub-branches. Drawing inspiration from negative feedback mechanisms in automatic control [Franklin *et al.*, 2002], FBQuant feeds the sub-branch weights back into the main path, ensuring that the reconstructed weights remain inherently bounded by the quantization process (see Sec. 4.1). With this mechanism, FBQuant is able to fine-tune sub-branches with a limited amount of calibration data, and prevent overfitting. The pipeline is both straightforward and effective as illustrated in Fig. 3, and we provide rigorous mathematical proofs in Secs. 3 and 4. To address the latency challenge, we develop the CUDA kernel fusion implementation on the sub-branches. By reducing the repeated read and write operations in the constructed layers, this implementation saves 60% of extra inference time compared to conventional sub-branches.

Experiments demonstrate that FBQuant outperforms existing methods across various tasks and model families. On 3-bit Llama2-7B, FBQuant improves zero-shot accuracy by 1.2%. We further apply FBQuant to instruction-tuned versions of these models, showing consistent gains over other quantization techniques. Wall-clock time evaluations further reveal significant token throughput improvements over traditional implementations, while matching the latency of quantized models without sub-branches.

Our contributions are summarized as follows:

- **Feedback-Based Sub-Branch Optimization.** We introduce a novel sub-branch quantization pipeline, FBQuant, which feeds sub-branch signals back into the main path to establish more effective reconstruction objectives. This design inherently upper-bounds the reconstructed weights, thereby preventing overfitting to calibration noise.
- **Efficient CUDA Kernel Integration.** We develop a tailored CUDA kernel that fuses the sub-branch operations with the main path, significantly reducing latency by minimizing repeated reads and writes to inputs, intermediates, and outputs.
- **Superior Performance Across Models.** Extensive experiments demonstrate that FBQuant consistently improves perplexity and zero-shot accuracy over existing quantization approaches across a variety of model families and parameter sizes.

2 Related Works

Quantization Methods are commonly divided into two categories: Post-Training Quantization (PTQ) [Dong *et al.*, 2019; Liu *et al.*, 2023a] and Quantization-Aware Training (QAT) [Liu *et al.*, 2023b]. PTQ offers training-free solutions by reconstructing the quantized model after it has been fully trained. For example, AdaRound [Nagel *et al.*, 2020] optimizes the rounding direction inspired by a Hessian-induced reconstruction objective, while Hawq [Dong *et al.*, 2019] leverages Hessian information to identify weights sensitive to quantization. In contrast, QAT incorporates quantization during the training process, mitigating the accuracy loss associated with quantization. In the case of large language models (LLMs), PTQ is generally preferred for its simplicity and lower computational overhead. In this work, we focus on PTQ for LLMs and introduce FBQuant.

Weight-only Quantization focuses solely on quantizing model weights, reducing memory usage and speeding up memory-bound operations. For instance, [Dettmers *et al.*, 2022] introduce the INT8 weight quantization method. QLoRA[Dettmers *et al.*, 2024] proposes the NF4 data format to better align with LLM weight distributions. GPTQ[Frantar *et al.*, 2022] uses layer-wise quantization combined with Optimal Brain Compression[Frantar and Alistarh, 2022], leveraging inverse Hessian information to guide quantization process. AWQ[Lin *et al.*, 2024b] identifies salient weights by the corresponding activations and applies scaling techniques to protect them during quantization. OmniQuant[Shao *et al.*, 2023] refines quantization via clamping and rotation values learned per layer. However, these methods suffer from significant accuracy degradation under low-bit quantization. In contrast, FBQuant employs an orthogonal sub-branch mech-

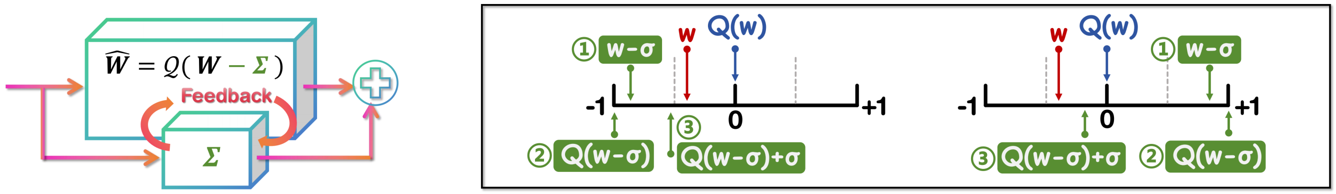

Figure 3: (Left) The main path incorporates feedback signals from the sub-branch to facilitate improved weight quantization, where $\hat{\mathbf{W}}$ represents the quantized weights in the main path, obtained via a quantizer $\mathcal{Q}(\cdot)$, and $\mathbf{\Sigma}$ denotes the weights in the sub-branch. (Right) Direct quantization of the original weights (red) maps them to the nearest quantization bins (blue). In contrast, the FBQuant method (green) applies a multi-step quantization approach, progressively adjusting the weights towards their original values in three stages.

anism that effectively compensates for quantization errors, leading to improved performance.

Weight-Activation Quantization handles both weights and activations. SmoothQuant [Xiao *et al.*, 2022] and Outlier Suppression [Wei *et al.*, 2022] achieve W8A8 quantization by balancing the quantization challenges of weights and activations. QServe [Lin *et al.*, 2024c] employs a W4A8 scheme, offering a trade-off between accuracy and speed. Quarot [Ashkboos *et al.*, 2024] and DuQuant [Lin *et al.*, 2024a] further enhance accuracy by introducing Hadamard rotation on weights and activations. RPTQ[Yuan *et al.*, 2023] and LLM-QAT[Liu *et al.*, 2023b] achieve W4A4 quantizaiton. However, weight-activation quantization generally faces insufficient support on general devices, making weight-only methods more applicable in practice.

Sub-branches Compensation introduces parallel pathways alongside the quantized layer to offset quantization errors. LoftQ [Li *et al.*, 2023a] directly decompose quantization errors using Singular Value Decomposition and construct low-rank residual paths correspondingly. CALDERA [Saha *et al.*, 2024] relies on quantized low-rank sub-branches that are fine-tuned to further reduce quantization errors, while EoRA [Liu *et al.*, 2024a] projects compression errors into the eigenspace of input activations, focusing on reconstructing the most impactful components. SVDQuant [Li *et al.*, 2024], originally developed for diffusion models but adaptable to LLMs, observes that high-rank components capture most of the outliers, leaving the remaining components simpler to quantize. However, these methods lack robust optimization strategies or employ suboptimal reconstruction objectives, often resulting in diminished performance or overfitting to calibration data noise. In contrast, our FBQuant introduces a feedback-driven optimization mechanism that effectively addresses the issue.

3 Motivation

This section discusses our insights on the ill-posed optimization of existing sub-branching methods (Sec. 3.1), and the inference delay induced by the sub-branches (Sec. 3.2).

3.1 Ill-posed Optimization

Sub-branch compensation reconstructs the weights of a layer can be mathematically represented as:

$$\mathbf{W}' = \mathbf{W}_{\mathcal{Q}} + \mathbf{\Sigma}, \tag{1}$$

where $\mathbf{W}$ is the original weight matrix, $\mathbf{W}_{\mathcal{Q}}$ is its quantized counterpart produced by quantizer $\mathcal{Q}$, and $\mathbf{\Sigma}$ is a low-rank linear projection derived from the sub-branch. Although this design aims at correcting quantization errors, directly optimizing $\mathbf{\Sigma}$ can lead to overfitting and dependence on calibration data. To see why, let's consider the layer-wise reconstruction objective:

$$\mathbf{\Sigma}^* = \arg\min_{\mathbf{\Sigma}} \mathcal{L}_1, \quad \text{subject to} \quad \text{rank}(\mathbf{\Sigma}) \leq r, \tag{2}$$

where

$$\begin{aligned} \mathcal{L}_1 &= \|(\mathbf{W} - \mathbf{W}')\mathbf{X}^\top\|_F \\ &= \|(\mathbf{W} - \mathbf{W}_{\mathcal{Q}} - \mathbf{\Sigma})\mathbf{X}^\top\|_F, \end{aligned} \tag{3}$$

$\|\cdot\|_F$ denotes the Frobenius norm, $\mathbf{X}$ represents layer inputs, and r is the rank constraint for the sub-branch. Letting $\mathbf{\Delta} = \mathbf{W} - \mathbf{W}_{\mathcal{Q}}$, assume $\mathbf{\Sigma}^*$ yields a minimal value ϵ_1 for $\mathcal{L}_1$, then

$$\begin{aligned} \epsilon_1 &= \|(\mathbf{\Delta} - \mathbf{\Sigma}^*)\mathbf{X}^\top\|_F \\ &= \text{tr}\left((\mathbf{\Delta} - \mathbf{\Sigma}^*)\mathbf{X}^\top\mathbf{X}(\mathbf{\Delta} - \mathbf{\Sigma}^*)^\top\right), \end{aligned} \tag{4}$$

where $\text{tr}(\cdot)$ denotes the trace operation. In addition, from the Singular Value Decomposition (SVD), $\mathbf{\Sigma}^*$ can be expanded in terms of the top-r singular vectors:

$$\mathbf{\Sigma}^* = \mathbf{U}_r\mathbf{S}_r\mathbf{V}_r. \tag{5}$$

We can subsequently consider an alternative solution $\mathbf{\Sigma}'$:

$$\mathbf{\Sigma}' = \mathbf{\Sigma}^* + \mathbf{\Sigma}_N, \quad \text{where} \quad \mathbf{\Sigma}_N = \mathbf{U}_r\mathbf{S}_r(\alpha\mathbf{N}_r), \tag{6}$$

with $\mathbf{N}_r$ sharing the same dimensionality as $\mathbf{V}_r$, and a scalar α. Because typical calibration data are limited, $\mathbf{X}^\top\mathbf{X}$ is positive semidefinite but not strictly full-rank [Huang *et al.*, 2024], implying there exists non-zero $\mathbf{N}_r$ orthogonal to $\mathbf{X}^\top\mathbf{X}$, that is,

$$\mathbf{N}_r\mathbf{X}^\top\mathbf{X} = \mathbf{0}. \tag{7}$$

Then,

$$\begin{aligned} \mathbf{\Sigma}_N\mathbf{X}^\top\mathbf{X} &= \mathbf{U}_r\mathbf{S}_r(\alpha\mathbf{N}_r\mathbf{X}^\top\mathbf{X}) \\ &= \mathbf{0}. \end{aligned} \tag{8}$$

Hence, the construction loss ϵ' which is achieved by $\mathbf{\Sigma}'$ is equal to ϵ_1:

$$\begin{aligned} \epsilon' &= \text{tr}\left((\mathbf{\Delta} - \mathbf{\Sigma}')\mathbf{X}^\top\mathbf{X}(\mathbf{\Delta} - \mathbf{\Sigma}')^\top\right) \\ &= \text{tr}\left((\mathbf{\Delta} - \mathbf{\Sigma}^*)\mathbf{X}^\top\mathbf{X}(\mathbf{\Delta} - \mathbf{\Sigma}^*)^\top\right) \\ &\quad - \text{tr}\left((\mathbf{\Delta} - \mathbf{\Sigma}^*)(\mathbf{\Sigma}_N\mathbf{X}^\top\mathbf{X})^\top\right) \\ &\quad - \text{tr}\left((\mathbf{\Sigma}_N\mathbf{X}^\top\mathbf{X}) \cdot (\mathbf{\Delta} - \mathbf{\Sigma}^* - \mathbf{\Sigma}_N)^\top\right) \\ &= \epsilon_1, \end{aligned} \tag{9}$$

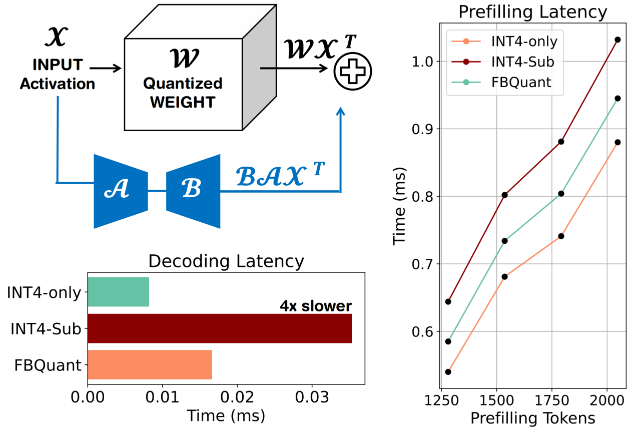

Figure 4: Macs and latency of the linear layer in Llama2-7B. (Up-left) The MACs introduced by the main path $\mathbf{WX}^\top$ and the sub-branch $\mathbf{BAX}^\top$ are $M_0 = b \times d \times d$ and $M_1 = 2 \times b \times r \times d$, respectively, where b is the batch size, r is the rank value, and d is the layer dimension. This results in $M_1/M_0 = 6.25\%$ additional MACs, when $r = 128$ and $d = 4096$. However, naively implementing this sub-branch significantly increases the latency by 20% when prefilling (right), and up to four times when decoding (bottom-left). FBQuant significantly mitigates the problem caused by the kernel fusion approach.

where gray terms are $\mathbf{0}$ due to orthogonality. This equation demonstrates that $\mathbf{\Sigma}'$ is also a valid solution. However, for any $w \in \mathbf{W}$, which is reconstructed by $\sigma' \in \mathbf{\Sigma}'$, the difference between the original and the reconstructed weights is:

$$\begin{aligned} |w - w'| &= \left|w - \big(\mathcal{Q}(w) + \sigma'\big)\right| \\ &= |w - \mathcal{Q}(w) - \sigma^* - \alpha\sigma_N|. \end{aligned} \tag{10}$$

With determined $\{w, \mathcal{Q}, \sigma^*\}$, the unbounded term $\alpha\sigma_N$ may significantly deviate the reconstructed weights, leading to less meaningful values, which highlights the potential for overfitting and performance degradation.

3.2 Delay Attribution of Sub-branches

We observe that inference latency can be significantly impacted by the inclusion of sub-branches. Theoretically, the computational overhead introduced by sub-branches constitutes only a small fraction of the total computation in the main path. However, in practice, it can lead to a substantial increase in inference latency. As illustrated in Fig. 4, consider a linear layer in the Llama2-7B model with a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$, a layer input and output dimension of $d = 4096$, and a sub-branch implemented by the LoRA [Hu *et al.*, 2021] approach with a rank value of 128. The inclusion of the sub-branch results in a 6.25% increase in Multiply-Accumulate Operations (MACs), but causes the decoding process to be four times slower inside the layer. This dramatic slowdown is primarily attributable to memory access bottlenecks. The sub-branch computations involve intensive access to input activations in the down projection (i.e., $\mathbf{A}$), multiple writes to intermediate results before the up projection (i.e., $\mathbf{B}$), and additional writes to the layer outputs.

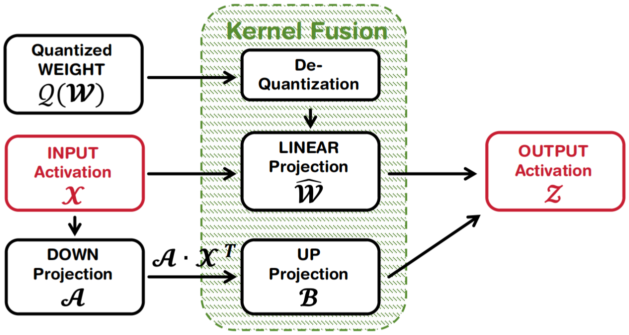

Figure 5: **Kernel Fusion.** We integrate the de-quantization and the linear projection in the main path, and up-projection in the sub-branch into the same kernel. The reduced number of kernels results in reduced kernel launch time. The integration reduces repeated writes to output activations.

4 Methodology

Building on the insights from Sec. 3, we propose FBQuant, which includes: (1) feedback integration to upper-bound reconstructed weights and prevent overfitting (see Sec. 4.1), and (2) a differentiability strategy ensuring gradient propagation for the sub-branch parameters to process layer-wise reconstruction (see Sec. 4.2). (3) Furthermore, we introduce an efficient CUDA kernel implementation to mitigate the extra latency induced by the sub-branch (see Sec. 4.3).

4.1 Feedback and Upper Bound

Conventional sub-branch methods overlook on how far the reconstructed weights can deviate from the original. To address this limitation, we introduce a novel feedback mechanism, hereafter referred to as **FBQuant**, which incorporates negative sub-weights into the quantization process. As illustrated in Fig. 3, the weights of the main path are $\mathcal{Q}(\mathbf{W} - \mathbf{\Sigma})$, and the weights of the sub-branch are $\mathbf{\Sigma}$, then, the feedback-based reconstructed weights $\mathbf{W}_F$ are redefined as:

$$\mathbf{W}_F = \mathcal{Q}(\mathbf{W} - \mathbf{\Sigma}) + \mathbf{\Sigma}. \tag{11}$$

Letting w and σ be elements of $\mathbf{W}$ and $\mathbf{\Sigma}$ respectively, the difference between the original and reconstructed weights is defined as:

$$\begin{aligned} |w - w_F| &= \left|w - \big(\mathcal{Q}(w - \sigma) + \sigma\big)\right| \\ &= |(w - \sigma) - \mathcal{Q}(w - \sigma)|. \end{aligned} \tag{12}$$

When $\mathcal{Q}(\cdot)$ functions as round-to-nearest, the deviation in Eq. (12) is bounded as:

$$|w - w_F| \leq s/2, \tag{13}$$

where s is the quantizer's scaling factor. Some toy examples are shown in Fig. 3 (left). Consequently, FBQuant naturally limits reconstructed-weight deviations, which is a critical property that prevents the sub-branch from overfitting calibration data noise.

4.2 Differentiability for Layer-wise Reconstruction

We fine-tune sub-branch weights based on layer-wise reconstruction. Before that, we must ensure the differentiability of the reconstruction objectives. Let's first look at the loss function derived by FBQuant:

$$\begin{aligned}\mathcal{L}_F &= \|\mathbf{W}\mathbf{X}^\top - \mathbf{W}_F\mathbf{X}^\top\|_F \\ &= \text{tr}\left(\mathbf{\Delta}_F\mathbf{X}^\top\mathbf{X}\mathbf{\Delta}_F^\top\right),\end{aligned} \tag{14}$$

where

$$\begin{aligned}\mathbf{\Delta}_F &= \mathbf{W} - \mathbf{W}_F \\ &= \mathbf{W} - \mathcal{Q}(\mathbf{W} - \mathbf{\Sigma}) - \mathbf{\Sigma}.\end{aligned} \tag{15}$$

Although Straight-Through Estimator (STE) [Bengio *et al.*, 2013] is commonly used to approximate the derivative of the quantizer as

$$\frac{\partial\mathcal{Q}(\mathbf{W})}{\partial\mathbf{W}} \approx \mathbf{I}, \tag{16}$$

it yields zero gradients for the sub-branch weights $\mathbf{\Sigma}$ in FBQuant, that is

$$\begin{aligned}\frac{\partial\mathcal{L}_F}{\partial\mathbf{\Sigma}} &= \frac{\partial\mathcal{L}_F}{\partial\mathbf{\Delta}_F}\cdot\frac{\partial\mathbf{\Delta}_F}{\partial\mathbf{\Sigma}} \\ &= (2\cdot\mathbf{\Delta}_F\mathbf{X}^\top\mathbf{X})\cdot(\mathbf{0} + \mathbf{I} - \mathbf{I}) \\ &= \mathbf{0}.\end{aligned} \tag{17}$$

To overcome this issue, we detach the feedback signal from the back-propagation graph, while allowing gradients to flow only through the sub-branch. This modification yields

$$\frac{\partial\mathbf{\Delta}_F}{\partial\mathbf{\Sigma}} = -\mathbf{I}. \tag{18}$$

Then, we obtain:

$$\frac{\partial\mathcal{L}_F}{\partial\mathbf{\Sigma}} = -2\cdot\mathbf{\Delta}_F\mathbf{X}^\top\mathbf{X}, \tag{19}$$

which enables optimizing $\mathbf{\Sigma}$ by gradient descent. In practice, we implement $\mathbf{\Sigma}$ using low-rank adapters [Hu *et al.*, 2021], such that $\mathbf{\Sigma} = \mathbf{B}\cdot\mathbf{A}$. The sub-branches are applied to the linear layers in LLMs, such as Query, Key, Value, and Out projections in attention blocks, as well as Down, Gate, and Up projections in feed-forward networks (FFNs). During the optimization process, calibration data is fed into the model and each layer is optimized using gradient descent to minimize the reconstruction loss in Eq. (14). More details of layer-wise reconstruction by FBQuant are described in algorithm 1.

4.3 Kernel Fusion

To address the latency challenge induced by sub-branches describe in Sec. 3.2, we propose the kernel fusion method, illustrated in Fig. 5. The reconstructed layer consists of four kernels: de-quantization and linear projection in the main path, as well as down-projection and up-projection in the sub-branch. We observe that while the down-projection requires relatively minimal time, the other operations dominate the overall latency. To mitigate this, we integrate weight de-quantization, activation-weight multiplication, and the up-projection into a single CUDA kernel. This reduces the total number of kernels from four to two, significantly lowering the kernel launch overhead. Additionally, in the fused kernels, the up-projection in the sub-branch shares the same output tensor as the linear projection in the main path. This optimization minimizes redundant writes, which would otherwise occur if the operations were handled separately. As a result, these modifications collectively reduce the additional inference delay caused by the sub-branches by 60%.

Algorithm 1 Layer-wise Reconstruction by FBQuant

```
Require: $\mathbf{X}_l$: input activations of the $l$-th layer
    $\mathbf{W}_l$: original weights of the $l$-th layer
    $\mathcal{Q}$: linear quantizer
    $\mathbf{\Sigma}_l$, $\mathbf{A}_l$, $\mathbf{B}_l$: $\mathbf{\Sigma}_l \leftarrow \mathbf{B}_l\mathbf{A}_l$, weights forming the low-rank sub-branch
    $r$: rank of the sub-branch satisfies rank($\mathbf{\Sigma}$) $\leq r$
Ensure: $\mathbf{A}_l'$, $\mathbf{B}_l'$: optimized low-rank sub-branch weights
$\mathbf{A}_l \leftarrow$ instantiate $\mathbf{A}_l \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$;
$\mathbf{B}_l \leftarrow$ instantiate $\mathbf{B}_l$ with $\mathbf{0}$;
for $l = 1 \rightarrow N$ do
    repeat
        obtain FP16 output:    $\mathbf{Z}_l \leftarrow \mathbf{W}_l\mathbf{X}_l^\top$
        obtain quantization output:
            $\mathbf{Z}_l' \leftarrow \mathcal{Q}(\mathbf{W}_l - \mathbf{B}_l\mathbf{A}_l)\mathbf{X}_l^\top + \mathbf{B}_l\mathbf{A}_l\mathbf{X}_l^\top$
        calculate reconstruction loss:    $\mathcal{L} \leftarrow \|\mathbf{Z}_l - \mathbf{Z}_l'\|_F$
        back propagation:
            $\frac{\partial\mathcal{L}}{\partial\mathbf{\Sigma}}$ with detached $\mathbf{W}_l$ and $\mathcal{Q}(\mathbf{W}_l - \mathbf{B}_l\mathbf{A}_l)$
        update $\mathbf{A}_l, \mathbf{B}_l$ via gradient descent
    until Convergence
    $\mathbf{A}_l', \mathbf{B}_l' \leftarrow \mathbf{A}_l, \mathbf{B}_l$
end for
return $\{\mathbf{A}_l', \mathbf{B}_l'\}_{l=1}^N$
```

5 Experiments

In this section, we present the experimental setup of models, baselines, datasets, metrics and implementation details in Sec. 5.1. Then, we demonstrate the perplexity and zero-shot accuracy of various quantization methos in Sec. 5.2, followed by the performance of instruction-tuned models and the wall-clock latency on real devices.

5.1 Experimental Setup

Models. We benchmark our methods using the model frameworks and checkpoints from HuggingFace [Jain, 2022], which includes Llama2 [Touvron *et al.*, 2023], Llama3 [Dubey *et al.*, 2024], and Qwen2.5 [Yang *et al.*, 2024] families, with parameter sizes ranging from 7 billion to 70 billion. Specifically, we use pre-trained versions to generate the main results as shown in Tabs. 1 and 2, while instruction-finetuned versions are utilized for the pairwise competition as shown in Fig. 6.

Baselines. We compare FBQuant against Round-To-Nearest (RTN), GPTQ [Frantar *et al.*, 2022], AWQ [Lin *et al.*, 2024b], OmniQuant [Shao *et al.*, 2023], CALDERA [Saha *et al.*, 2024], and SVDQuant [Li *et al.*, 2024]. GPTQ, AWQ, and OmniQuant are implemented using their publicly released codebase. For CALDERA and SVDQuant, we follow their papers and codebases to produce the results, as

Method	W Bit	Group	Llama2-7B	Llama2-13B	Llama2-70B	Llama3-8B	Llama3-70B	Qwen2.5-7B
FP	16	-	5.12	4.57	3.12	5.75	2.97	6.39
RTN	4	128	5.27	4.69	3.25	6.29	3.63	6.72
GPTQ	4	128	5.27	4.67	3.23	7.31	3.41	6.60
AWQ	4	128	5.23	4.65	3.20	6.11	3.25	6.62
OmniQuant	4	128	5.23	4.65	3.19	6.15	3.18	6.60
CALDERA	4	128	5.26	4.69	3.20	6.26	3.23	7.14
SVDQuant	4	128	5.30	4.71	3.22	6.39	3.14	6.64
FBQuant	4	128	**5.18**	**4.61**	**3.15**	**5.89**	**3.05**	**6.52**
RTN	3	128	6.08	5.18	3.73	10.74	12.14	11.50
GPTQ	3	128	5.96	5.07	3.69	8.66	5.04	7.90
AWQ	3	128	5.82	4.98	3.56	7.63	4.39	7.31
OmniQuant	3	128	5.78	4.96	3.53	7.86	4.12	7.23
CALDERA	3	128	5.84	5.07	3.71	9.64	4.78	10.20
SVDQuant	3	128	6.90	5.54	3.69	10.73	4.30	8.57
FBQuant	3	128	**5.59**	**4.86**	**3.42**	**6.78**	**3.77**	**6.92**

Table 1: Comparison of perplexity scores on the WikiText2 validation dataset. Lower values indicate better performance. Results for GPTQ, AWQ, and OmniQuant were obtained using their publicly released codebases. Results for CALDERA and SVDQuant were derived from their publicly released codebases with necessary revisions. The best results are highlighted in bold. Our FBQuant approach consistently demonstrates superior performance.

CALDERA only support Lattice quantizer and SVDQuant is originally designed for diffusion models. Additionally, we include unquantized models using the float16 datatype as a baseline for a fair comparison.

Datasets and Metrics. Following previous works [Frantar *et al.*, 2022; Lin *et al.*, 2024b], we employ 128 samples with a sequence length of 2048 in the subset of WikiText2 [Merity *et al.*, 2016] training data for calibration. The perplexity results are tested on the WikiText2 validation set. The zero-shot evaluation is conducted using the open-source toolkit, i.e., Language Model Evaluation Harness [Gao *et al.*, 2024], which has been utilized by other baselines. The evaluation datasets include Arc-Challenge [Clark *et al.*, 2018], Arc-Easy [Clark *et al.*, 2018], HellaSwag [Zellers *et al.*, 2019], MMLU [Hendrycks *et al.*, 2021], PIQA [Bisk *et al.*, 2020], WinoGrande [Sakaguchi *et al.*, 2019], and BoolQ [Wang *et al.*, 2019]. We report the averaged accuracy.

Implementation Details. All experiments are conducted using A100 and RTX 3090 GPUs. Both the A100 and 3090 GPUs are utilized for optimizing the sub-branches, while only the 3090 GPU is used for testing latency, as it is commonly available for personal use. In the main results, we set the rank parameter to 128. The total number of optimization epochs is set to 20. A group size of 128 is used in all quantization methods. Sub-branches are integrated into all linear layers in LLMs, such as Query, Key, Value, and Out projections in Attention blocks, as well as Down, Gate, and Up projections in Feed-Forward Networks.

5.2 Results and Analysis

Performance Comparison. Tab. 1 and Tab. 2 present the evaluation results for perplexity on WikiText2, and zero-shot accuracy on seven public benchmarks, across different quantization methods, tested on various LLM architectures and model sizes. The quantization methods evaluated include RTN, GPTQ, AWQ, and OmniQuant, which rely on clamping and rotation techniques; as well as CALDERA, SVDQuant, and FBQuant, which adopt sub-branch approaches. Our FBQuant achieves state-of-the-art results in both perplexity and zero-shot accuracy across various models. Specifically, for the 3-bit Llama3-8B model, FBQuant reaches a perplexity of 6.78, marking a significant improvement of 0.85 over the second-best method AWQ. For the Llama2-7B model, FBQuant attains a zero-shot accuracy of 64.68%, outperforming OmniQuant by 1.20%. Three key observations can be made from the results: (a) 4-bit quantization generally offers good performance, while 3-bit quantization remains a little gap compared to the floating-point models. (b) New models present greater challenges for quantization. For instance, the 3-bit Llama3-8B model poses significant challenges for AWQ, which suffers a substantial degradation in perplexity by 1.88. (c) Existing sub-branch approaches occasionally produce poor performance, as observed with CALDERA on 3-bit version of Llama3-8B, Llama3-70B, and Qwen2.5-7B, as it relies on the ill-posed reconstruction objective as explained in Sec. 3. Besides, SVDQuant performs poorly on 3-bit Llama3-8B, as it only focuses on the weight outliers, ignoring the layer output error.

Quantization of Instruction-tuned Models. Most real-world applications are powered by instruction-tuned models, motivating us to benchmark the performance of our method on the instruction-tuned versions of the corresponding LLMs. Following the experiment setting in [Lin *et al.*, 2024b; Shao *et al.*, 2023], we conduct pairwise comparisons among AWQ, OmniQuant, CALDERA, SVDQuant, and FBQuant under 3-bit settings. We use the GPT-4 evaluation protocol [Chiang *et al.*, 2023] to assess performance on the Vicuna benchmark [Chiang *et al.*, 2023], which comprises 80 questions. The results of the comparisons are reported as win, tie, and loss percentages. A higher percentage of wins and ties indicates better performance, highlighting the superior results achieved by FBQuant. To negate position bias [Zheng *et al.*,

Method	W Bit	Group	Llama2-7B	Llama2-13B	Llama2-70B	Llama3-8B	Llama3-70B	Qwen2.5-7B
FP	16	-	66.62	70.45	76.27	72.79	80.22	74.25
RTN	4	128	64.17	69.83	75.87	71.52	78.67	71.55
GPTQ	4	128	64.78	69.93	75.85	66.12	79.51	72.09
AWQ	4	128	66.08	70.14	76.07	71.78	79.13	73.15
OmniQuant	4	128	65.75	69.85	76.10	71.81	79.37	73.41
CALDERA	4	128	65.88	68.37	76.01	70.09	79.24	68.78
SVDQuant	4	128	65.12	69.94	76.18	71.42	77.85	73.85
FBQuant	4	128	**66.45**	**70.05**	**76.20**	**72.55**	**79.74**	**74.03**
RTN	3	128	62.17	68.33	73.97	61.51	68.61	67.21
GPTQ	3	128	59.68	67.16	73.39	63.63	77.19	70.73
AWQ	3	128	63.31	68.74	74.82	67.83	77.69	70.84
OmniQuant	3	128	63.48	67.96	74.55	67.97	77.81	70.98
CALDERA	3	128	62.10	66.13	73.56	63.48	77.17	64.45
SVDQuant	3	128	57.06	64.49	74.61	60.38	78.47	70.79
FBQuant	3	128	**64.68**	**69.11**	**75.77**	**68.87**	**78.99**	**72.16**

Table 2: Comparison of zero-shot accuracy on seven benchmarks as described in the experimental setup. The table shows the average accuracy across all tasks, with the highest score highlighted in bold. Our FBQuant approach consistently demonstrates superior performance. More details are provided in the Appendix.

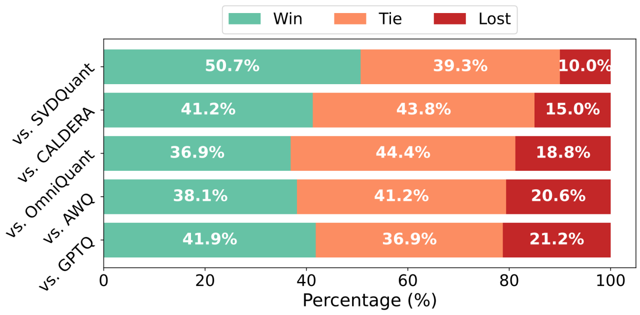

Figure 6: Quantization of Instruction-tuned Models. We use different quantization methods on the Llama3-8B-Chat model to conduct the experiments. FBQuant demonstrates consistent strength against other quantization methods.

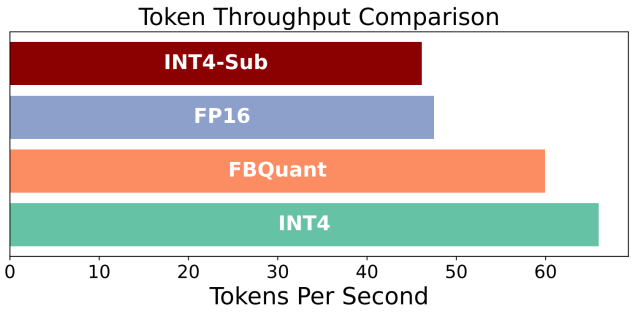

Figure 7: Token throughput of the Llama2-7B model using FP16, INT4-Sub, INT4 and INT4-FBQuant tested on RTX 3090 GPU. Experiments were conducted with a batch size of 1. The rank parameter for INT4-Sub and FBQuant was set to 128. The token throughput is measured by prefilling 256 and decoding 64 tokens.

2023], we also test pairs in exchanged position, totaling 160 trails per competition. As shown in Fig. 6, in the Llama3-8B-Chat model, our FBQuant demonstrates best performance against other quantization methods. For instance, FBQuant achieves 79.3% win-tie rate against to AWQ, and 90.0% win-tie rate against to SVDQuant.

Wall-clock Latency on Real Devices. We evaluate the token throughput, measured in tokens per second (tk/s), of the FBQuant kernels on the RTX 3090 GPU, and compare it to floating-point (FP16), INT4, and INT4 with conventional sub-branch (INT4-Sub) implementations. As shown in Fig. 7, INT4-Sub exhibits a significant slowdown, achieving only 46 tk/s, slightly slower than FP16, which achieves 48 tk/s. In contrast, FBQuant demonstrates a substantial improvement against baselines, achieving 61 tk/s and significantly increasing token throughput.

6 Conclusions

In this paper, we introduced FBQuant, a novel feedback-based optimization approach that addresses key challenges in the sub-branching quantization for LLMs. By incorporating a feedback mechanism inspired by automatic control systems, FBQuant effectively optimizes sub-branches without overfitting to calibration data, ensuring robust reconstruction of quantized weights. Additionally, we developed an efficient CUDA kernel fusion implementation to tackle the latency overhead introduced by sub-branches, significantly reducing inference delays caused by memory access bottlenecks. Our experiments demonstrated that FBQuant consistently outperforms existing quantization techniques across various tasks and model families on LLMs, achieving superior accuracy and perplexity while improving inference efficiency. Specifically, FBQuant enhances zero-shot accuracy for 3-bit models, such as Llama2-7B, by 1.2%, and achieves significant throughput improvements in wall-clock evaluations. These results highlight the practical utility of FBQuant for on-device deployment of LLMs, where computational and memory constraints are critical.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2022YFB4400900, in part by the Strategic Industries and Key Technologies Project of Jiangsu Province under Grant BE2023020-3 and the Basic Research Program of Jiangsu Province under Grant BK20243042.

References

[Ashkboos *et al.*, 2024] 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. *arXiv preprint arXiv:2404.00456*, 2024.

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

[Bisk *et al.*, 2020] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[Chiang *et al.*, 2023] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[Clark *et al.*, 2018] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[Dettmers *et al.*, 2022] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

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

[Dong *et al.*, 2019] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019.

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

[Franklin *et al.*, 2002] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, 2002.

[Frantar and Alistarh, 2022] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

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

[Gao *et al.*, 2024] 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. A framework for few-shot language model evaluation, 07 2024.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[Hu *et al.*, 2021] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[Huang *et al.*, 2024] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291*, 2024.

[Jain, 2022] Shashank Mohan Jain. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pages 51–67. Springer, 2022.

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

[Li *et al.*, 2023b] Yuan Li, Yixuan Zhang, and Lichao Sun. Metaagents: Simulating interactions of human behaviors for llm-based task-oriented coordination via collaborative generative agents. *ArXiv*, abs/2310.06500, 2023.

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

[Lin *et al.*, 2024a] 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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

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

[Liu *et al.*, 2023a] Yijiang Liu, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. Noisyquant: Noisy bias-enhanced post-training activation quantization for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20321–20330, 2023.

[Liu *et al.*, 2023b] Zechun Liu, Barlas Oğuz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *ArXiv*, abs/2305.17888, 2023.

[Liu *et al.*, 2024a] Shih-Yang Liu, Huck Yang, Chein-Yi Wang, Nai Chit Fung, Hongxu Yin, Charbel Sakr, Saurav Muralidharan, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, et al. Eora: Training-free compensation for compressed llm with eigenspace low-rank approximation. *arXiv preprint arXiv:2410.21271*, 2024.

[Liu *et al.*, 2024b] 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. *arXiv preprint arXiv:2405.16406*, 2024.

[Merity *et al.*, 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[Nagel *et al.*, 2020] Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. *ArXiv*, abs/2004.10568, 2020.

[Saha *et al.*, 2024] Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J. Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *ArXiv*, abs/2405.18886, 2024.

[Sakaguchi *et al.*, 2019] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. An adversarial winograd schema challenge at scale. 2019.

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

[Talebirad and Nadiri, 2023] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *ArXiv*, abs/2306.03314, 2023.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Wang *et al.*, 2019] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*, 2019.

[Wei *et al.*, 2022] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *ArXiv*, abs/2209.13325, 2022.

[Xiao *et al.*, 2022] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *ArXiv*, abs/2211.10438, 2022.

[Yang *et al.*, 2024] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[Yuan *et al.*, 2023] Zhihang Yuan, Lin Niu, Jia-Wen Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. Rptq: Reorder-based post-training quantization for large language models. *ArXiv*, abs/2304.01089, 2023.

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

[Zheng *et al.*, 2023] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.