Mitigating Quantization Errors Due to Activation Spikes in GLU-Based LLMs

Anonymous ACL submission

Abstract

Modern large language models (LLMs) have established state-of-the-art performance through architectural improvements, but still require significant computational cost for inference. In an effort to reduce the inference cost, post-training quantization (PTQ) has become a popular approach, quantizing weights and activations to lower precision, such as INT8. In this paper, we reveal the challenges of activation quantization in GLU variants (Shazeer, 2020), which are widely used in feed-forward network (FFN) of modern LLMs, such as LLaMA family. The problem is that severe local quantization errors, caused by excessive magnitudes of activation in GLU variants, significantly degrade the performance of the quantized LLM. We denote these activations as *activation spikes*. Our further observations provide a systematic pattern of activation spikes: 1) The activation spikes occur in the FFN of specific layers, particularly in the early and late layers, 2) The activation spikes are dedicated to a couple of tokens, rather than being shared across a sequence. Based on our observations, we propose two empirical methods, Quantization-free Module (QFeM) and Quantization-free Prefix (QFeP), to isolate the activation spikes during quantization. Our extensive experiments validate the effectiveness of the proposed methods for the activation quantization, especially with coarse-grained scheme, of latest LLMs with GLU variants. In particular, our methods enhance the current quantization techniques that struggle with the activation spikes.[1]

1 Introduction

Large language models (LLMs) have become a key paradigm in natural language processing, accelerating the release of variations within the community (Zhao et al., 2023; Wei et al., 2022). Furthermore, latest LLMs establish state-of-the-art performance by training with increased scale, as well as by adopting architectural improvements such as GLU (Shazeer, 2020), RoPE (Su et al., 2024), GQA (Ainslie et al., 2023), and MoE (Jiang et al., 2024). Especially, GLU (Gated Linear Unit) variants (e.g., SwiGLU, GeGLU) has been adopted in the most of modern LLM architectures (e.g., LLaMA family (Touvron et al., 2023a)), due to training efficiency (Shazeer, 2020; Narang et al., 2021). Although LLMs broaden foundational capabilities in natural language tasks and potential for various applications, billions of parameters in the large models impose considerable computational costs on end users in practice. To reduce GPU memory requirements and accelerate inference speed, post-training quantization (PTQ) offers an affordable solution by quantizing weights and activations into a lower precision (e.g., INT8) without a need for expensive retraining steps (Jacob et al., 2018; Gholami et al., 2022; Nagel et al., 2021). However, recent studies have revealed that large magnitude values at certain coordinates exist in the activations of LLMs, which are often called outliers, posing a key challenge in activation quantization (Dettmers et al., 2022; Xiao et al., 2023a; Ahmadian et al., 2023; Wei et al., 2023). Another line of works attempts to explain the role of outlier values in the attention mechanism (Bondarenko et al., 2024; Sun et al., 2024).

In this paper, we present our discovery that the GLU architecture in the feed-forward network (FFN) generates excessively large activation values, which are responsible for significant local quantization errors. Specifically, we observe that these problematic activation values occur in specific linear modules and are dedicated to a couple of tokens, which will be discussed in Section 3. To distinguish the excessive GLU activations from the outliers, we refer to them as *activation spikes*. In light of our observations, we propose two empirical methods to mitigate the impact of activation spikes on quan-

[1]Code is available at https://anonymous.4open.science/r/74FC/.

tization: Quantization-free Module (QFeM) and Quantization-free Prefix (QFeP). QFeM aims to partially exclude quantization for linear layers (or modules) where large quantization errors occur, instead of quantizing the entire linear modules in the LLM. By scoring the extent of scale disparity, QFeM selects linear modules to exclude. On the other hand, QFeP identifies the prefix that triggers activation spikes and preserves its context as a key-value (KV) cache, thereby preventing the recurrence of activation spikes in subsequent tokens. It is noteworthy that both QFeM and QFeP rely on calibration results to capture activation spikes in advance, without any modifications to the target LLM. Thus, our methods can be integrated into any existing quantization methods.

In our comprehensive experiments, we demonstrate that recently released LLMs incorporating GLU variants struggle with activation spikes when applying activation quantization. Consequently, the proposed methods, QFeM and QFeP, substantially enhance the performance of the primitive quantization method, the round-to-nearest (RTN) method. Furthermore, we observe that current outlier alleviation methods (Xiao et al., 2023a; Wei et al., 2023) and state-of-the-art quantization methods (Zhao et al., 2024; Shao et al., 2023) are exposed to the activation spikes and benefit from our proposed methods. Compared to the strong baseline of fine-grained activation quantization (Yao et al., 2022), our methods show competitive performance, achieving reduced latency and memory footprint.

In summary, the contributions of our work are as follows:

- We find that the GLU architecture in modern LLMs systematically generates excessive activation values, which are responsible for significant performance degradation when quantizing the entire model and input tokens.

- Based on our observations, we propose two empirical methods, QFeM and QFeP, which effectively exclude the activation spikes during quantization, with negligible computational overhead and compatibility with any existing quantization techniques.

- Our experiments validate the detrimental impact of the activation spikes on activation quantization, while proposed methods consistently enhance the quantization performance.

2 Related Works

Outlier Values in LLMs. Since the advent of LLMs (Brown et al., 2020; Zhang et al., 2022) rooted in the GPT (Radford et al., 2019), recent studies (Dettmers et al., 2022; Xiao et al., 2023a; Ahmadian et al., 2023) have tackled the existence of outlier values in LLMs. These outliers exhibit a large magnitude of values at the shared dimensions of hidden states across tokens. More recently, Bondarenko et al. (2024); Sun et al. (2024) explain that the outliers attribute to the vertical pattern in the attention mechanism (Xiao et al., 2023b; Kovaleva et al., 2019), which influences the performance of LLMs. In particular, Sun et al. (2024) claims a different type of outlier existing in the hidden states of specific tokens. However, prior studies merely focus on the superficial hidden states between the decoder layers. Our work provides a module-level investigation where quantization is applied practically, focusing on different LLM architectures.

Post-training Quantization for LLMs. Post-training quantization (PTQ) refers to the quantization of a neural network model to low precision, such as INT8, without additional parameter updates (Jacob et al., 2018; Gholami et al., 2022). Especially for LLMs, this approach cost-effectively achieves inference with low memory usage and faster inference latency by quantizing the weights and activations used in matrix multiplication (e.g., linear layer). However, because of the challenges in activation quantization of LLMs, many recent works are mainly focused on the weight-only quantization (Frantar et al., 2022; Kim et al., 2023b; Lin et al., 2023; Chee et al., 2024; Yao et al., 2023; Dettmers et al., 2023; Shao et al., 2023). Otherwise, the activation quantization faces inherent outliers, which hinder accurate quantization by reducing representation resolution. To address this challenge, Dettmers et al. (2022) proposes a mixed-precision quantization method where the outlier dimensions are computed in high precision. Xiao et al. (2023a); Wei et al. (2023); Shao et al. (2023) approach migration of scale from activation to weights to alleviate the scale of outlier activations. Most similar to our work, Liu et al. (2024); Son et al. (2024) utilize prompt to improve the quantization accuracy. We highlight that our work uncovers how the prefix works on GLU activations and proposes an efficient prefix length which is sufficient for handling activation spikes, while they focus on preserving attention sink (Xiao et al., 2023b) pattern.

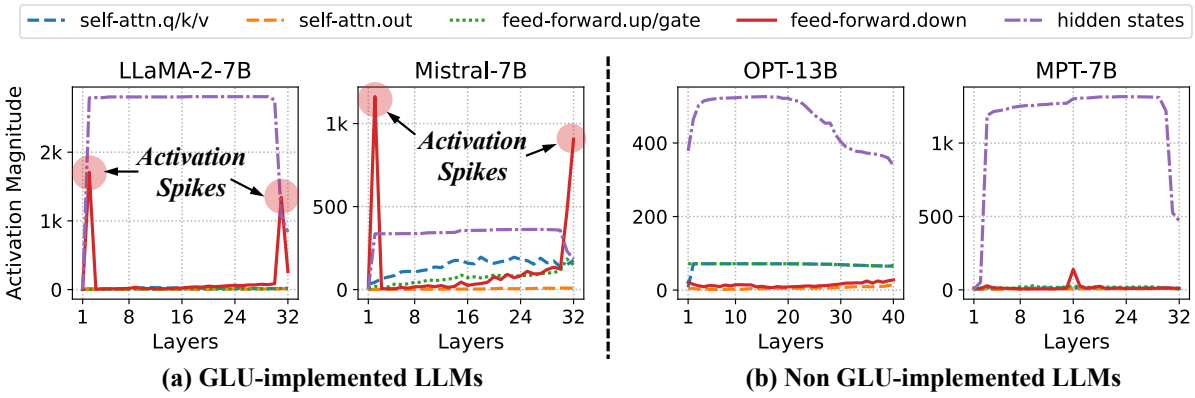

Figure 1: Calibration results on GLU-implemented and non GLU-implemented LLMs. We present the maximum magnitudes of input activations for each linear modules and layer-wise hidden states. For more results on different LLMs, see Appendix A.2, A.3.

3 Activation Spikes: Excessive Magnitude of GLU Activations

For clarity, "hidden states" refer to the output tensor of a transformer layer (or block), while "input activations" or "activations" denote the input tensor of a linear layer (or module) in the remain of this paper. Recent work (Sun et al., 2024) has investigated a novel type of outlier existing in the hidden states across modern LLMs. Although these outliers of hidden states play a crucial role in the attention mechanism (Sun et al., 2024; Bondarenko et al., 2024; Xiao et al., 2023b), their relationship with input activations for quantization has not been fully explored. Importantly, because recent LLMs adopt Pre-LN (Xiong et al., 2020; Baevski and Auli, 2018), which normalizes hidden states before self-attention and feed-forward network (FFN) blocks, the scale of hidden states does not reflect the scale of input activations within the transformer block. Therefore, we focus on the input activations fed into each linear module within the transformer block to connect to activation quantization. Specifically, we examine the four linear (projection) layers: `query` (parallel to `key` and `value`), `out`, `up` (parallel to `gate`), and `down` modules. For details of Pre-LN transformer, please see Appendix F.2.

3.1 Existence of Activation Spikes in GLU Variants

To analyze the input activations, we employ a calibration method, which is used to estimate the quantization factors such as scale and zero-point. For the calibration data, we use 512 samples randomly collected from the C4 (Raffel et al., 2020) training dataset. Afterwards, we feed each sample into the LLM and monitor each hidden state and input activation through the decoder layers. To estimate the scale factor, we use absolute maximum value. The tested LLMs are listed in Table 8.

GLU-implemented LLMs exhibit activation spikes at specific layers. In Figure 1a, we display the calibrated scale factors for the LLMs that implement GLU variants (e.g., SwiGLU, GeGLU). Across models, we observe a shared pattern of scale from the results. Within the early and late layers, the `down` modules in the FFN show noticeable magnitudes of input activations. Note that these input activations are derived from the *Hadamard Product* within GLU. Thus, the GLU variants generate activation spikes at the specific layers. Interestingly, we notice a high correlation between the emergence of activation spikes and intermediate hidden states of large scale. This indicates that the FFN contributes to amplifying the hidden states via the addition operation in the residual connection (He et al., 2016). Once the magnitude of the hidden states is exploded, it persists through layers until encounter the activation spikes at late layers.

Non GLU-implemented LLMs show modest scale distribution. Figure 1b illustrates the calibration results for LLMs with the original feed-forward implementation in Transformer (Vaswani et al., 2017). We observe that the LLMs continue to generate the large-scale hidden states, regardless of the GLU implementation. This corresponds to the observations in Sun et al. (2024). More importantly, our module-level results elaborate that the scale of hidden states is not transferable to the input activations of inner linear modules. Instead, we reveal that GLU variants are associated with the hidden states and generate activation spikes. This clarifies the quantization challenge of the GLU-

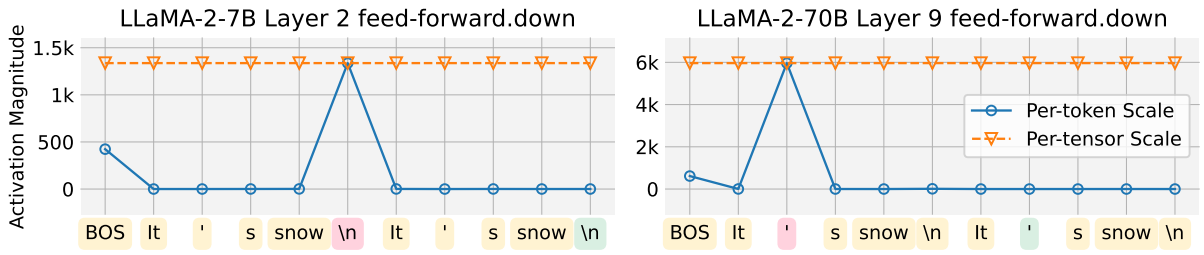

Figure 2: Token-wise scales in a specific layer with an activation spike. When quantizing the input activations using a per-tensor scale, the scale of the activation spike dominates the scales of the other tokens. The red (green) box indicates a token with (without) an activation spike. For more examples, see Appendix F.1.

implemented LLMs concentrated in the early and late layers. Because excessive scales of activation spikes have the potential to hinder the accurate quantization, we conduct an in-depth analysis to better understand these activation spikes in the following sections.

3.2 Token-level Scale Analysis within Activation Spikes

In the previous section, we observed the excessive scale of the input activations derived from GLU activation. When quantizing the input activations, the variance of input activation scales for each token affects the quantization performance (Yao et al., 2022). To delve into the disparity between token-wise scales in the activation spikes, we unroll them through the sequence of tokens. Figure 2 illustrates the individual input activation scales where the activation spike appears. Given a token sequence, the large magnitudes of input activations are observed in a couple of tokens, such as the BOS token, newline (`\n`), and apostrophe (`'`). These specific tokens coincide with the observations of Sun et al. (2024), which suggests that such tokens exhibit massive values in the hidden states. Thus, the activation spike is associated with the process of assigning a special role to these tokens in later transformer layers. However, the excessive scale of specific token hinders the estimation of scale factor for the other tokens, such as in per-tensor quantization. Additionally, the largest scale is dedicated to the first instance of the specified token, while the following usage exhibits a modest scale. This phenomenon makes the quantization more complicated, as the activation spikes dynamically occur depending on the current input sequence.

3.3 Effect of Quantization on Activation Spikes

We explore the impact of local quantization errors caused by activation spikes on LLM outputs.

Model	Perplexity(↓)			
	FP16	Top 4	Middle 4	Bottom 4
LLaMA-2-7B	7.37	**11.77**	7.38	7.40
LLaMA-2-13B	6.84	**15.09**	6.84	6.84
Mistral-7B	8.35	**69.45**	8.35	8.36
Gemma-7B	10.85	**85.83**	10.94	10.87

Table 1: Perplexity of partial activation quantization of LLMs. We select only four modules to quantize, ranked by $r^{(m)}$. The results of MSE are provided in Appendix E.2.

To identify the layers where activation spikes occur, we utilize a ratio between the maximum and median values of the token-wise input activation scales, instead of using the maximum scale value alone. The max-median ratio for linear layer m can be formulated as $r^{(m)} = \frac{\max(\mathbf{S}^{(m)})}{\text{median}(\mathbf{S}^{(m)})}$, where $S^{(m)}$ represents the token-wise input activation scales incoming to module $m \in M$. This max-median ratio captures the extent to which maximum scale dominate the other token scales. For comparison, we choose the activation quantization targets as the top-4, middle-4, and bottom-4 modules, based on the max-median ratio in descending order. Then, we evaluate the perplexity and mean-squared error (MSE) using the calibration dataset. Here, the MSE is calculated for the last hidden states between the original (FP16) and partially quantized LLM. As shown in Table 1, quantization on the top-4 rated modules solely degrades the LLM performance by significant margins, while the other cases exhibit negligible performance changes. We consider these quantization-sensitive input activations (*inter alia* activation spikes) to be the quantization bottleneck, which, in this paper, refers to the quantization error caused by outliers.

Furthermore, the activation spikes are conditioned on the specific context of the input sequence as discussed in Section 3.2. Altogether, such dynamic bottlenecks must be handled with caution to enhance the quantization performance of LLMs.

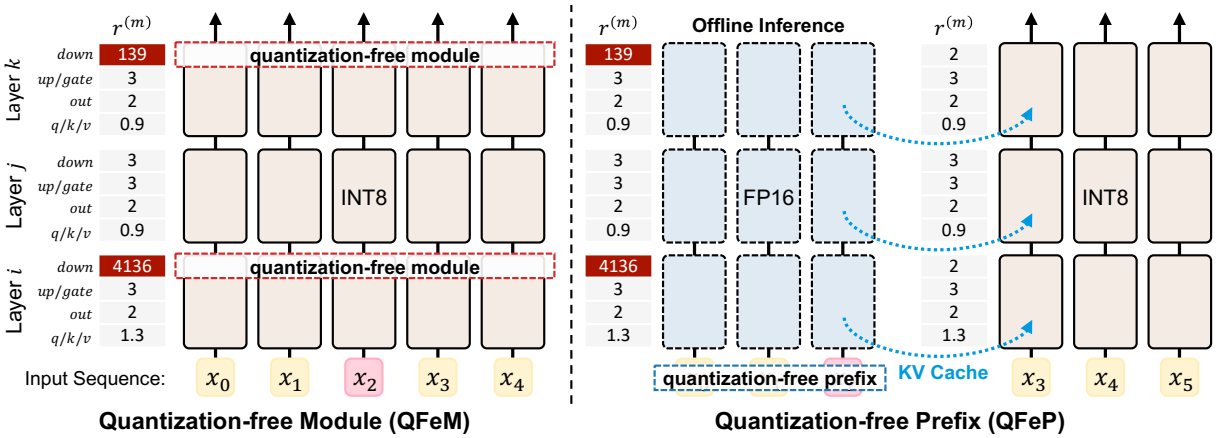

Figure 3: Overview of QFeM and QFeP. (Left): QFeM excludes the modules whose $r^{(m)}$ is larger than the threshold α from quantization. (Right): QFeP computes in advance the prefix of activation spikes and utilizes solely their KV cache during the quantization phase, effectively preventing further activation spikes in subsequent sequences.

4 Mitigating Quantization Quality Degradation Based on the Observation

To address the quantization bottleneck, our approach is based on the deterministic occurrence patterns of activation spikes. First, we utilize the observation that bottlenecks occur at a few specific layers. This implies that naive full quantization of LLMs is affected by these bottlenecks. Second, we exploit the phenomenon that the activation spike is derived from the first occurrence of specific tokens. Thus, the planned occurrence prevents recurrence in the subsequent and possibly future tokens. In the following sections, we propose two methods inspired the above insights.

4.1 Quantization-free Module (QFeM)

In the full quantization of LLM, all linear layers within the LLM are quantized. Among these linear layers, we propose omitting the quantization of input activations for linear layers where significant quantization errors are caused by activation spikes. To be noted, increasing the number of unquantized modules exhibits a trade-off between the inference latency and the model performance. Thus, determining which module should be quantized (or left unquantized) is crucial to retain the efficacy of quantization. Here, we use the max-median ratio $r^{(m)}$ and define a set of unquantized modules, denoted as M_{unq}, where the ratio $r^{(m)}$ of each linear layer is larger than threshold α. For instance, all linear layers in M are quantized if $\alpha = \infty$. For clarity, we treat sibling linear layers, such as query-key-value, as a single linear layer. To control the impact of activation quantization only, we leave the weight parameters in unquantized linear layers as INT8 and dequantize them into FP16 during matrix multiplication with the incoming activations, operating as weight-only quantization.

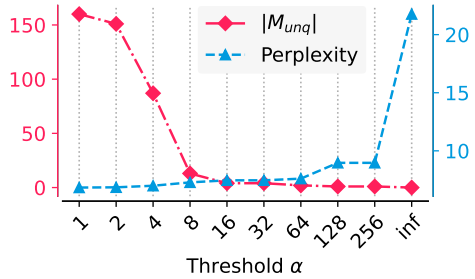

Figure 4: Trade-off between perplexity (stands for performance) and $|M_{unq}|$ (stands for latency) according to the threshold α for LLaMA-2-13B model.

Optimizing the threshold α. To calculate the activation scale ratio for each linear layer, we first gather token-wise input activation scales from the calibration examples discussed in Section 3.1. Exceptionally, for FFN experts in the mixture of experts (MoE) architectures like the Mixtral model (Jiang et al., 2024), calibration is performed separately. After determining these ratios, we use binary search to set the threshold value α, balancing inference latency and performance degradation. As a metric, we assess performance through perplexity measured on the same calibration examples. For example, the relationship between threshold value α and its impact on performance is depicted in Figure 4, demonstrating how full quantization can degrade performance. Rather than fully quantizing, we identify an optimal threshold by finding the intersection of two performance curves; in Figure 4, this threshold is approximately 16. Details on the QFeM implementation are provided in Table 2.

4.2 Quantization-free Prefix (QFeP)

Orthogonal to the QFeM, we propose Quantization-free Prefix (QFeP) that mitigates the quantization errors by precomputing the prefix (or short prompt) corresponding to activation spikes. This method is based on the observations presented in Section 3.2, which indicate that significant quantization errors result from the overestimated scale factor of the *first instance* within the restricted token set. Inspired by this occurrence pattern of activation spikes, we aim to construct a prefix which stabilizes the quantization scale factor of the tokens that come after the prefix. In other words, once the prefix is fixed at the beginning, the activation spikes consistently occur within the prefix. Afterward, we employ key-value (KV) caching mechanism to process the activation spikes in advance. In practice, KV cache is utilized to optimize the decoding speed of causal language models by storing precomputed key and value states of the previous tokens (Pope et al., 2023; Ott et al., 2019). This approach provides a bypass of the quantization including activation spikes, while preserving the context of prefix through the KV cache. The KV cache for the prefix is precomputed once through the offline inference of LLM without quantization. Then, this KV cache is exploited in the quantization phases, such as calibration or dynamic quantization, even for quantized inference. The process of QFeP is illustrated in Figure 3. We verified that our prefix design is effective in mitigating the quantization errors through a prefix ablation study (Appendix D).

Prefix Search. To form a prefix of explicit activation spike, we first identify candidate token that represent the activation spike at the linear layer with the highest max-median ratio $r^{(m)}$. For instance, the candidate token can be apostrophe (') token for LLaMA-2-70B model, as highlighted in red in Figure 2. Once the candidate token is identified, we search the middle context token for between the BOS token and the candidate token in the prefix. This middle context provides dummy context, which is required to activate the candidate token. To find the middle context, we design a template $[B, T_1, C_1, T_2, C_2]$ where B, T_i, and C_i denote the BOS token, context token, and candidate token in the vocabulary V, respectively. Then, we select the context token T where C_1 triggers an activation spikes, while later instance of the same token C_2 does not. When the context token for the activation spikes is varied, we choose the token that maximizes the activation scale ratio between the C_1 and C_2. Finally, we prepare the KV cache for searched prefix of $[B, T, C]$. Note that the latter sequence in the template can be replaced with sequences from dataset instead of repetition.

Model	Prefix	α	$\lvert M_{unq}\rvert / \lvert M\rvert$
LLaMA-2-7B	[BOS] all .	6.68	17 / 128
LLaMA-2-13B	[BOS] then ,	12.91	6 / 160
LLaMA-2-70B	[BOS] I '	9.16	25 / 320
Mistral-7B	[BOS] how \n	49.00	3 / 128
Mixtral-8x7B	[BOS]). \n	4.03	191 / 608
SOLAR-10.7B	[BOS] a 1	6.48	11 / 192
Gemma-7B	[BOS] . Più	10.65	5 / 112
LLaMA-3-8B	[BOS] - nd	6.64	6 / 128
LLaMA-3-70B	[BOS] and ,	78.37	3 / 320

Table 2: Specifications for QFeM and QFeP used in experiments. $|M|$ denotes the total number of linear layers in the LLM, and $|M_{unq}|$ represents the number of unquantized layers for QFeM.

Implementation Details. During the prefix search phase, we exploit the calibration dataset used in Section 3.1. For the candidate tokens, we consider the tokens with the top three largest input activation magnitudes. Then, we search for the middle context token among top 200 most frequent tokens in the calibration dataset, which is the subset of the vocabulary V. Finally, with the search result, we prepare the KV cache for the target model in FP16 precision. Exceptionally, for the Mixtral (Jiang et al., 2024) model, we use the scale of output hidden states instead of input activations, as the tokens are divided sparsely in a mixture of experts architecture. Table 2 presents the searched prefix.

5 Experiments

5.1 Experimental Setup

Models. Our proposed methods, QFeM and QFeP, aim to mitigate the quantization bottleneck, which is discussed in Section 3.3, caused by the activation spikes, especially in the GLU variants. To validate the efficiency proposed methods, we tested publicly released LLMs that were implemented with GLU, according to their paper and source code. We recognize recent LLMs, including LLAMA-2-{7B, 13B, 70B} (Touvron et al., 2023b), LLaMA-3-{7B, 70B} (Dubey et al., 2024), Mistral-7B (Jiang et al., 2023), Mixtral-8x7B (Jiang et al., 2024), SOLAR-10.7B (Kim et al., 2023a), and Gemma-7B (Team et al., 2024), utilize the GLU architecture. The LLMs with original FFN are not covered, as they suffer from the existing outliers rather than

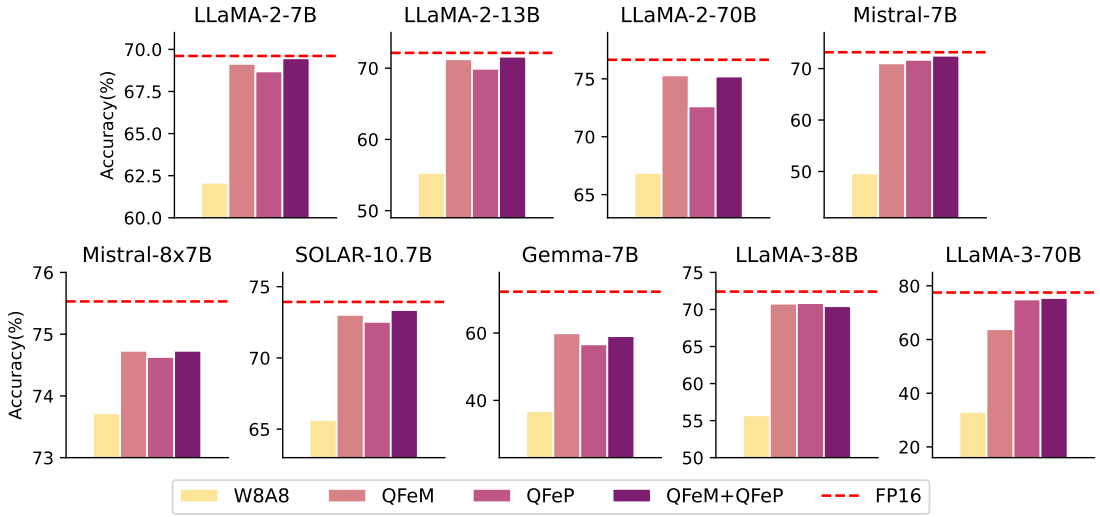

Figure 5: The average accuracy of zero-shot evaluation on GLU-implemented LLMs. FP16 denotes the original model precision, and W8A8 denotes the model quantized to INT8 for both weights and activations. Most models recover significantly compared to W8A8, with performance close to FP16. We provide full results of LLaMA-2 models in Table 9.

activation spikes. All models are sourced from the huggingface-hub[2] repository.

Quantization. In the experiments, we quantize both the input activations and the weights of linear layers for INT8 matrix multiplication operations. Note that in Table 2, $|M|$ denotes the total number of linear modules targeted for quantization. In these linear layers, we opt for dynamic per-tensor quantization as the quantization scheme of input activations, and per-channel quantization for weights, respectively. Regarding both input activations and weights, we symmetrically quantize the range using the absolute maximum value as the scale estimation function. For comparison, we use FP16 and per-token activation quantization (Yao et al., 2022) as baselines. We refer the reader to Appendix B for Batch Matrix-Multiplication (BMM) quantization, which involves quantizing tensors in the self-attention.

Evaluations. We evaluate the quantized LLMs with two metrics: zero-shot evaluation accuracy and perplexity. For zero-shot evaluation, we use the four datasets: PIQA (Bisk et al., 2020), LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2019). We utilize the lm-evaluation-harness library (Gao et al., 2023) to evaluate zero-shot tasks. To measure perplexity, we use the WikiText-2 (Merity et al., 2016) dataset. In all cases, we use the [BOS] token as the starting token for each input sequence by default.

[2]https://huggingface.co/models

5.2 Main Results

We report the evaluation results of quantization on GLU-implemented LLMs in Figure 5. Compared to FP16 precision, quantizing both weights and activations (W8A8) degrades the overall performance. The results demonstrate that our proposed methods resolve the activation spikes and, surprisingly, restore the performance of the W8A8 close to that of FP16. It is worth noting that, because QFeM and QFeP are targeted to address the activation spikes, they improve the model performance at comparable levels. This indicates that the activation spikes present a direct cause of the significant decrease in quantization performance. Because the proposed methods are orthogonal, the performance slightly increases when incorporating both QFeM and QFeP compared to applying them individually.

Our comprehensive model analysis offers deeper insights into the architecture. The Mixtral model showcases robustness against performance degradation. This indicates that the mixture of experts architecture, which assigns the MLP experts to specific tokens, helps mitigate the impact of activation spikes. On the other hand, addressing the activation spikes is not a sufficient complement for the Gemma model compared to other models. We attribute this to the choice of activation function among GLU variants; specifically, Gemma uses GeGLU, while other models employ SwiGLU.

5.3 Combining Outlier Alleviation Methods

While our method focuses on the activation spikes, the inherent outlier values in the input activations

Model	#Bits	Method	Per-token Quant.			Per-tensor Quant.		
			Base	QFeM	QFeP	Base	QFeM	QFeP
LLaMA-2-7B (FP16: 5.268)	W8A8	SQ	5.296	**5.288**	5.302	9.907	**5.534**	5.715
	W8A8	OSP	5.293	**5.288**	5.303	38.490	**5.493**	5.642
	W4A4	OSP	48.199	**16.151**	44.635	23491	**4772**	10940
LLaMA-2-13B (FP16: 4.789)	W8A8	SQ	4.809	**4.808**	4.818	34.869	**5.118**	6.551
	W8A8	OSP	4.813	**4.812**	4.819	5.148	**5.099**	5.144
	W4A4	OSP	21.037	**11.535**	11.860	9340	**8680**	9362

Table 3: Perplexity on Wikitext-2 for outlier alleviation methods with QFeM and QFeP under different activation quantization granularities. Activation quantization is applied at two levels: fine-grained (per-token) and coarse-grained (per-tensor). We employs per-channel granularity for weight quantization. For low-bit quantizations (W6A6 and W4A4), we adopt asymmetric scale estimation.

remain. Here, we combine the prior outlier alleviation methods, such as SmoothQuant (SQ) (Xiao et al., 2023a) and OutlierSuppressionPlus (OSP) (Wei et al., 2023), to further improve the quantization error. Table 3 demonstrates the evaluation results of applying the outlier alleviation methods solely and combining them with our methods. We find that there are cases where the alleviation method fails to recover the performance when quantizing the activations with per-tensor scheme.[3] This indicates that alleviating the outlier scales, including the activation spikes, is challenging. Under 4-bit quantization, we observe that QFeM and QFeP are efficient even when per-token quantization is applied. Similarly, we confirm this increase in our analysis of state-of-the-art quantization methods in Appendix C.

5.4 Computational Cost Analysis

The proposed methods require additional resources to evict the activation spikes. Therefore, we analyze the computational costs of the methods and compare them in various schemes. For comparison, we evaluate different activation quantization schemes: dynamic per-token, dynamic per-tensor, and static per-tensor, denoted as AQ1, AQ2, and AQ3, respectively. Figure 6 illustrates that while the fine-grained scheme (AQ1) shows a negligible accuracy drop, its counterparts (AQ2, AQ3) degrade due to the quantization bottleneck. However, by applying our methods, the coarse-grained schemes achieve a competitive performance gain. For example, the combination of AQ2 and QFeM demonstrates the performance close to the AQ1 but with faster latency and lower memory consumption (Appendix E.1). The results signify that addressing

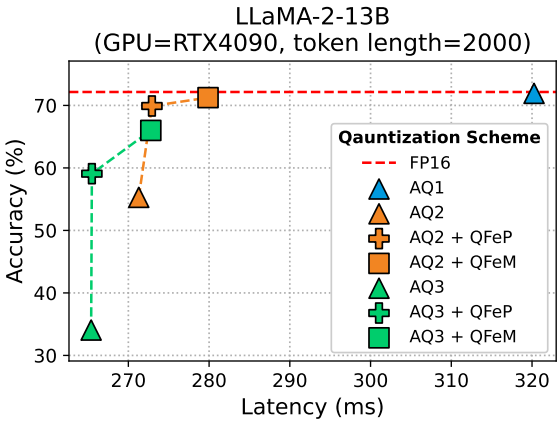

Figure 6: Accuracy-latency comparison of different activation quantization schemes: dynamic per-token (AQ1), dynamic per-tensor (AQ2), and static per-tensor (AQ3). The case of LLaMA-2-70B is provided in Figure 9.

the quantization bottleneck is important to accelerate the inference latency with coarser granularity. Specifically, the naive static quantization (AQ3), the fastest scheme, exhibits a significant decline. We hope that our work contributes to the future works, which address the remaining challenges in static quantization.

6 Conclusion

We explore the quantization challenge of GLU activations for modern LLMs. We find that the GLU variants generate excessive activation scales, which cause significant quantization bottlenecks at the specific layers. Based on the systematic generation pattern of the activation spikes, we propose methods that address the spikes in a layer-wise (QFeM) and token-wise manner (QFeP). In the experiments, we confirm that the proposed methods effectively resolve the quantization bottlenecks and result in a large performance gain. We expect that our work sheds light on the potential challenges in future studies regarding quantization and facilitates the development of efficient LLM systems.

[3] In original papers (Xiao et al., 2023a; Wei et al., 2023), the activations of LLaMA models are quantized using only a dynamic per-token scheme.

7 Limitations

Hardware Compatibility. For QFeM, we opted to certain modules unquantized where where it is difficult to quantize due to activation spikes. As a result, these linear modules are restricted to operations in full precision (FP16). To overcome this limitation, we propose an alternative solution: using fine-grained schemes specifically for the QFeM modules, rather than leaving them unquantized. Since activation spikes occur at certain tokens, per-token quantization (or even finer quantization) guarantees the quantization accuracy for these modules.

Non GLU-implemented LLMs. As discussed in Section 3.1, the activation spikes mainly occur in GLU activations. In our preliminary experiments, we observed that the performance gain of QFeM and QFeP is negligible for non GLU-implemented LLMs. However, in recent years, the outlier problem for non GLU-implemented LLMs has been identified, and there are many successful solutions such as SmoothQuant (Xiao et al., 2023a). We strongly believe that our work reveals a novel quantization challenge that has emerged with the advancement of LLMs and contributes to future researches by providing a better understanding of activation spikes.

References

Arash Ahmadian, Saurabh Dash, Hongyu Chen, Bharat Venkitesh, Zhen Stephen Gou, Phil Blunsom, Ahmet Üstün, and Sara Hooker. 2023. Intriguing properties of quantization at scale. *Advances in Neural Information Processing Systems*, 36:34278–34294.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. 2024. Stable lm 2 1.6 b technical report. *arXiv preprint arXiv:2402.17834*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2024. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36.

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

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. 2024. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36.

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

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

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

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

Leo Gao, 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. 2023. A framework for few-shot language model evaluation.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer.

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. 2023a. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2023b. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Ruikang Liu, Haoli Bai, Haokun Lin, Yuening Li, Han Gao, Zhengzhuo Xu, Lu Hou, Jun Yao, and Chun Yuan. 2024. Intactkv: Improving large language model quantization by keeping pivot tokens intact. *arXiv preprint arXiv:2403.01241*.

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

Javaheripi Mojan and Bubeck Sébastien. 2023. Phi-2: The surprising power of small language models.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*.

Sharan Narang, Hyung Won Chung, Yi Tay, Liam Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. 2021. Do transformer modifications transfer across implementations and applications? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5758–5773, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

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

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

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Seungwoo Son, Wonpyo Park, Woohyun Han, Kyuyeun Kim, and Jaeho Lee. 2024. Prefixing attention sinks can mitigate activation outliers for large language model quantization. *arXiv preprint arXiv:2406.12016*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. 2024. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1648–1665, Singapore. Association for Computational Linguistics.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023a. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023b. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

Zhewei Yao, Cheng Li, Xiaoxia Wu, Stephen Youn, and Yuxiong He. 2023. A comprehensive study on post-training quantization for large language models. *arXiv preprint arXiv:2303.08302*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.

A Additional Calibration Results

In this section, we provide details of LLMs when performing calibration, which is the step during quantization where the FP16 ranges are computed (Appendix A.1), and additional calibration results (Appendix A.2, A.3).

A.1 Detailed Specification of LLMs

In Section 3.1, we have performed the calibration method on various LLMs. We observe the calibration results by categorizing based on the presence of GLU in the LLMs. Table 8 shows the detailed structures of the LLMs. We refer notations for feed-forward implementiation from Shazeer (2020). In the case of GLU-implemented LLMs, which is LLaMA-2, LLaMA-3, Mistral, Mixtral, SOLAR, StableLM-2, and Gemma, most models have SwiGLU for FFN activation, while only Gemma has GeGLU. On the other hand, in non GLU-implemented LLMs, most of them utilize GeLU for FFN activation, with the exception of OPT, which uses ReLU.

A.2 Other Calibration Results on GLU-implementation

Figure 7 and 13 show the calibration result examples for various GLU-implemented LLMs that are not shown in the models in Figure 1a. In most GLU-implemented LLMs, we observe that the input activations have large values near the first and last layers. Unlike the typical GLU-implemented LLM architecture, Mixtral is composed of 8 feed-forward blocks in the single FFN, containing multiple gate linear units (Jiang et al., 2024). According to this structure, we can observe that one of the gates spikes in value in Figure 7.

A.3 Other Calibration Results on Non GLU-implementation

Figure 14 shows the calibration result examples for various non GLU-implemented LLMs that were not shown in the models in Figure 1b. There are no activation spikes on non GLU-implemented LLMs.

B BMM Quantization

To achieve faster inference latency, BMM operations in the self-attention also can be computed as INT8 operation (Xiao et al., 2023a). This requires a quantization on the query, key, and value states including the cached context. Because activation spikes produce a large magnitude of latent values,

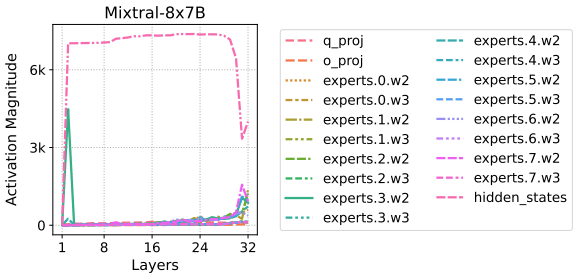

Figure 7: Calibration results on Mixtral-8x7B.

it is important to confirm the extent of quantization errors from KV quantization. This confirmation is necessary to gain advantages from BMM quantization. In Table 4, we examine the impact of BMM quantization on the W8A8 and QFeM. Regardless of the BMM quantization, the QFeM method consistently improves the quantization bottleneck. For example, the 13B and 70B models maintain their performance, while the 7B model shows a slight decrease. However, this decrease appears to be due to inherent quantization errors rather than a quantization bottleneck from activation spikes. As a result, we confirm that our QFeM method effectively improves the overall performance even in the BMM quantization scenario.

Model	Method	BMM Quantization No	BMM Quantization Yes
7B	W8A8	62.08%	61.66%
	+QFeP	68.69%	68.30%
13B	W8A8	55.29%	55.43%
	+QFeP	69.91%	69.77%
70B	W8A8	66.87%	66.75%
	+QFeP	72.62%	72.69%

Table 4: BMM quantization results.

C Low-bit Activation Quantization

In Section 5.3, we discussed the efficacy of the QFeM and QFeP for outlier alleviation methods. To concretize the the compatibility of our proposed methods, we provide additional experiment results for recent state-of-the-art (SOTA) quantization techniques, Atom (Zhao et al., 2024) and OmniQuant (Shao et al., 2023). Note that we follow their original implementation and parameter settings (e.g., fine-grained group quantization in Atom) during evaluation. Table 5 illustrates the results for SOTA methods for perplexity of WikiText-2 (Merity et al., 2016) and C4 (Raffel et al., 2020) datasets. As expected, our methods are compatible with SOTA quantization techniques and enhance their performance, especially for OmniQuant. For

Method	LLaMA-2-7B		LLaMA-2-13B	
	Wiki2	C4	Wiki2	C4
Atom (W4A4)	5.710	7.601	5.081	6.878
+QFeM	5.634	7.538	**5.071**	**6.854**
+QFeP	5.685	7.493	5.089	6.872
+QFeM+QFeP	**5.607**	**7.442**	5.073	6.855
OmniQuant (W4A4)	14.208	19.005	10.416	14.103
+QFeM	9.483	13.569	**9.499**	12.348
+QFeP	11.926	15.796	10.808	13.584
+QFeM+QFeP	**8.818**	**12.244**	9.901	**12.294**

Table 5: Evaluation of state-of-the-art activation quantization methods with QFeM and QFeP.

Atom, we observe that fine-grained group quantization has the potential to address activation spikes. By using this level of granularity, one can quantize the problematic linear modules for QFeM, rather than leaving them unquantized.

D Prefix Ablation Study

For the QFeP, we designed a length-three prefix for the KV cache, including the BOS token, context token, and extra token for activation spike. Because the KV cache consumes the capacity of the pretrained sequence position, it raises a question about the length of the prefix. Therefore, we conduct ablation study for different prefixes for the KV cache. For the prefixes, we prepare random, BOS only, and both QFeP without and with the context token. Notably, the BOS cache corresponds to the $IntactKV_{[B]}$ (Liu et al., 2024). We illustrate the results of ablation study in Figure 8. In all cases, the random prefix showcases the lowest performance. While the KV cache with the BOS token demonstrates inconsistent performance, our QFeP consistently shows significant improvement. Importantly, the results imply that the sufficient prefix for the models exhibits differences. However, we emphasize that our KV design for QFeP shows improvements by large margins across all models. Beyond our QFeP, $IntactKV_{[P]}$ (Liu et al., 2024) extends the BOS token to the system prompt for supervised fine-tuned models and CushionCache (Son et al., 2024) update prefixes through quantization-aware prefix tuning.

E Supplementary Experiment Results

E.1 Memory Footprint

In Table 6, we record the maximum memory footprint of QFeM and QFeP. For QFeP, the additional memory is consistently required for the preserved

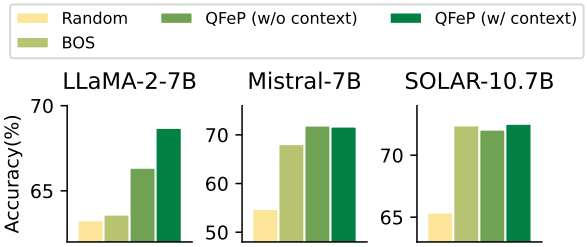

Figure 8: Prefix ablation. Y-axis represents averaged accuracy of four zero-shot tasks.

KV cache. However, this memory overhead is much smaller than that used in the fine-grained quantization (AQ1), as QFeM utilizes only three tokens for the cache. Contrary to QFeP, QFeM shows inconsistent memory utilization. For example, the 7B model with QFeM exhibits memory usage similar to AQ2, while the 70B model with QFeM incur additional consumption for a sequence length of 1K. This is attributed to the use of W8A16 for the unquantization modules in QFeM. To tailor the memory usage or inference speed, an alternative strategy can be utilized for QFeM, such as applying fine-grained activation quantization to the unquantization modules instead of using W8A16.

Method	SeqLen	
	1K	2K
LLaMA-2-7B		
AQ1	8185MiB	9516MiB
AQ2	8148MiB	9474MiB
+QFeP	8149MiB	9478MiB
+QFeM	8148MiB	9474MiB
LLaMA-2-70B		
AQ1	67756MiB	69037MiB
AQ2	67648MiB	68820MiB
+QFeP	67651MiB	68822MiB
+QFeM	67838MiB	68819MiB

Table 6: Memory footprint.

E.2 Results of Partial Activation Quantization

Extending Section 3.3, we provide MSE evaluation of partial activation quantization in Table 7.

Model	MSE(↓)		
	Top 4	Middle 4	Bottom 4
LLaMA-2-7B	**1908.80**	1.03	12.90
LLaMA-2-13B	**4762.11**	0.91	10.38
Mistral-7B	**218.60**	0.02	0.18
Gemma-7B	**213.93**	1.60	1.07

Table 7: MSE of partial activation quantization of LLMs. We select only four modules to quantize, ranked by $r^{(m)}$.

E.3 Additional Results for Inference Cost

We provide additional inference cost comparison for LLaMA-2-70B model in Figure 9.

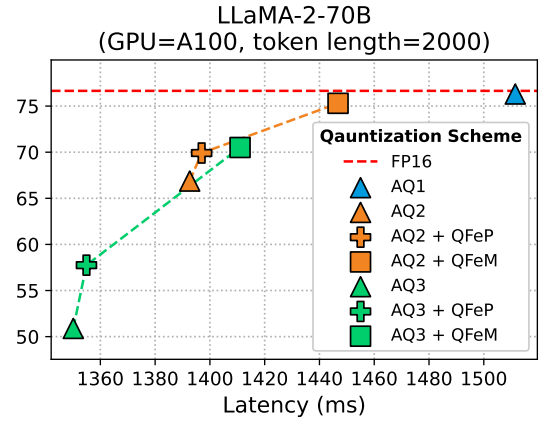

Figure 9: Accuracy-latency comparison for LLaMA-2-70B model.

F Miscellaneous

F.1 Additional Results for Token-level Scale Analysis

We provide additional results for token-level scale analysis (Section 3.2). In Figure 10 and Figure 11, the token for the activation spikes next to the BOS token does not exhibit the excessive activation scale.

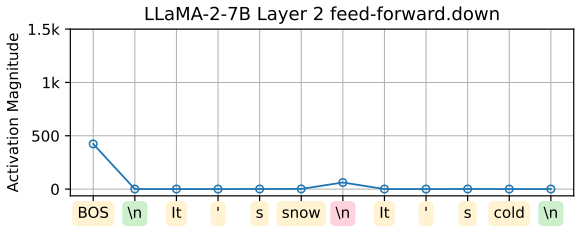

Figure 10: Token-wise scales analysis for LLaMA-2-7B. The newline token next to the BOS token does not exhibit the activation spikes.

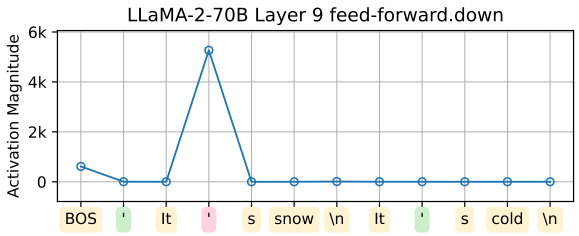

Figure 11: Token-wise scales from the unrolled activation spike of LLaMA-2-70B. The apostrophe token next to the BOS token does not exhibit the activation spikes.

F.2 Transformer Architecture

In Figure 12, we illustrate the Pre-LN transformer architecture and each sub-modules. We highlight with the same color the linear modules that accept identical input activations. Note that the hidden states are normalized before forwarding into the query and up linear modules.

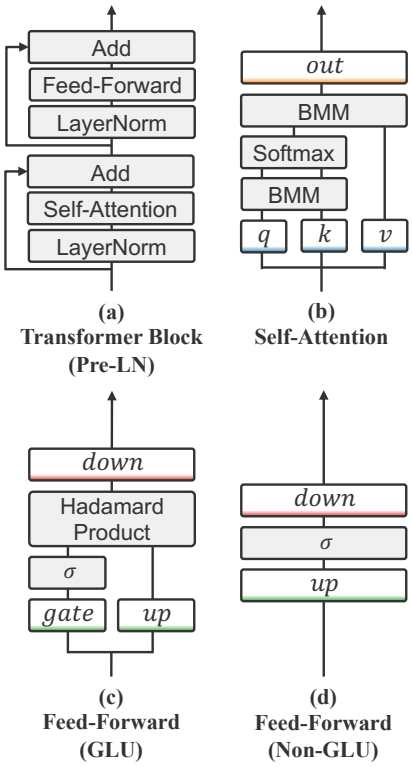

Figure 12: An illustration of Pre-LN transformer block and its sub-modules. Two feed-forward implementation, GLU and Non-GLU, are visualized in (c) and (d) respectively. In feed-forward network, σ denotes non-linear activation function, such as GeLU. We highlight the linear modules where input activations are quantized.

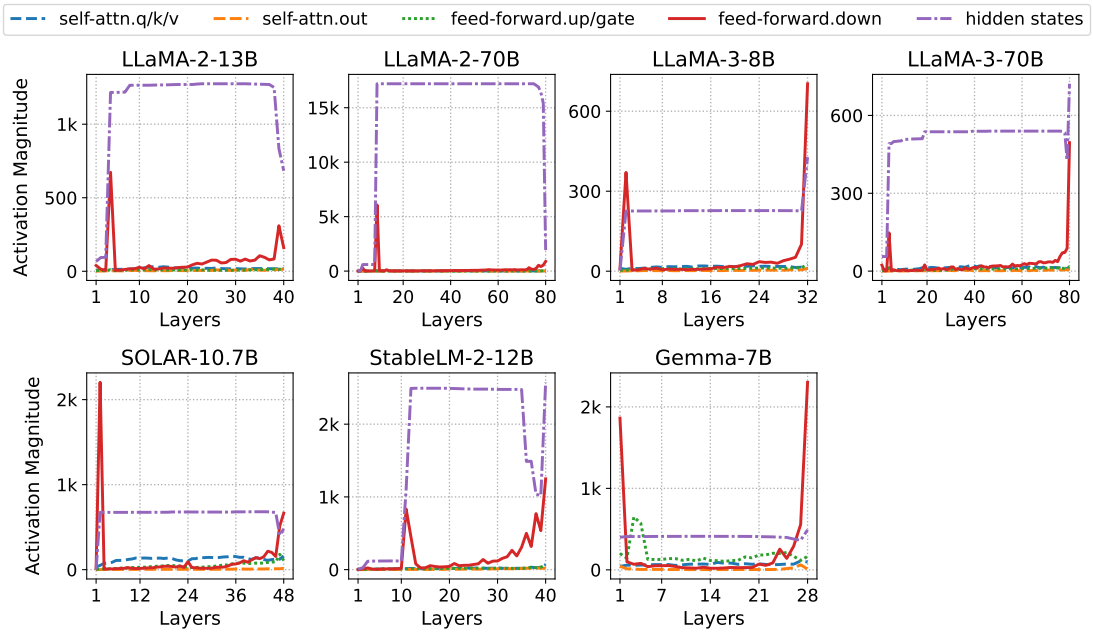

Figure 13: Calibration results on GLU-implemented LLMs.

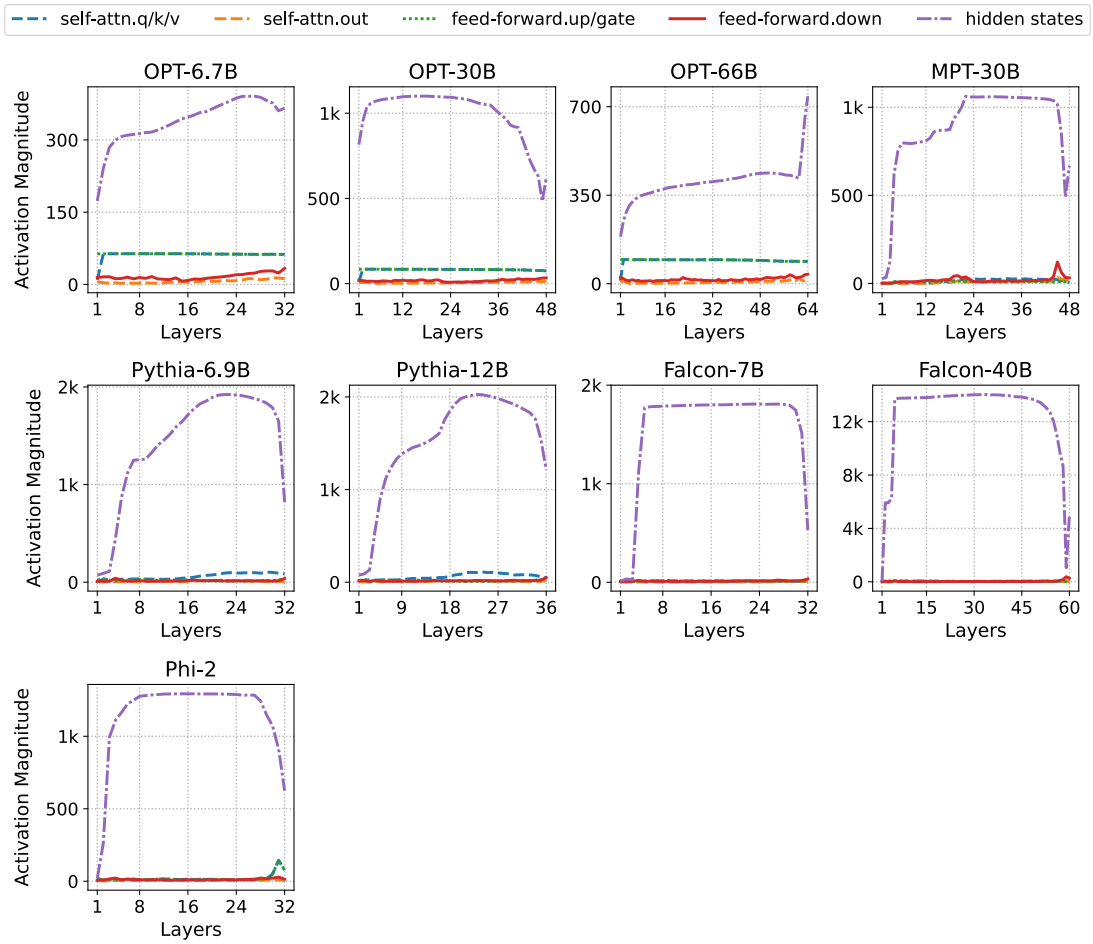

Figure 14: Calibration results on Non GLU-implemented LLMs.

Model	Size	FFN Activation	Normalization	PE	Vocabulary Size
GLU-implemented LLMs:					
LLaMA-2 (Touvron et al., 2023b)	7B, 13B, 70B	SwiGLU	RMSNorm	RoPE	32000
LLaMA-3 (Dubey et al., 2024)	8B, 70B	SwiGLU	RMSNorm	RoPE	128256
Mistral (Jiang et al., 2023)	7B	SwiGLU	RMSNorm	RoPE	32000
Mixtral (Jiang et al., 2024)	8x7B	SwiGLU	RMSNorm	RoPE	32000
SOLAR (Kim et al., 2023a)	10.7B	SwiGLU	RMSNorm	RoPE	32000
StableLM-2 (Bellagente et al., 2024)	12B	SwiGLU	LayerNorm	RoPE	100352
Gemma (Team et al., 2024)	7B	GeGLU	RMSNorm	RoPE	256000
Non GLU-implemented LLMs:					
OPT (Zhang et al., 2022)	6.7B, 13B, 30B, 66B	ReLU	LayerNorm	Learned	50272
MPT (Team, 2023)	7B, 30B	GeLU	LayerNorm	ALiBi	50432
Pythia (Biderman et al., 2023)	6.9B, 12B	GeLU	LayerNorm	RoPE	50432, 50688
Falcon (Almazrouei et al., 2023)	7B, 40B	GeLU	LayerNorm	RoPE	65024
Phi-2 (Mojan and Sébastien, 2023)	2.7B	GeLU	LayerNorm	RoPE	51200

Table 8: Architecture specification of LLMs. We categorize them into two groups depending on whether GLU is implemented in the FFN. All LLMs in the table use Pre-LN for the LayerNorm position.

Method	WikiText-2 (ppl↓)	PIQA (acc↑)	LAMBADA (acc↑)	HellaSwag (acc↑)	WinoGrande (acc↑)	Avg (acc↑)
LLaMA-2-7B						
FP16	5.268	78.18%	73.67%	57.13%	69.46%	69.61%
W8A8	8.634	72.80%	62.27%	49.57%	63.69%	62.08%
+QFeM	5.758[**-2.876**]	78.02%	73.86%	56.32%	68.35%	69.14%[**+7.06**]
+QFeP	5.758[**-2.876**]	76.44%	73.57%	55.55%	69.22%	68.69%[**+6.61**]
+QFeM+QFeP	5.573[**-3.061**]	77.86%	74.58%	56.05%	69.38%	69.47%[**+7.39**]
LLaMA-2-13B						
FP16	4.789	79.49%	76.54%	60.20%	72.38%	72.15%
W8A8	34.089	70.13%	49.66%	42.65%	58.72%	55.29%
+QFeM	5.241[**-28.848**]	77.58%	75.68%	59.13%	72.61%	71.25%[**+15.96**]
+QFeP	6.000[**-28.089**]	77.53%	73.94%	57.23%	70.96%	69.91%[**+14.62**]
+QFeM+QFeP	5.126[**-28.963**]	78.51%	75.86%	59.44%	72.61%	71.61%[**+16.32**]
LLaMA-2-70B						
FP16	3.218	81.45%	79.45%	65.29%	80.43%	76.65%
W8A8	8.055	74.05%	70.27%	55.21%	67.96%	66.87%
+QFeM	3.830[**-4.225**]	81.23%	77.66%	64.15%	78.14%	75.30%[**+8.43**]
+QFeP	6.007[**-2.048**]	77.64%	73.26%	63.40%	76.16%	72.62%[**+5.75**]
+QFeM+QFeP	3.708[**-4.347**]	81.23%	77.82%	64.65%	77.11%	75.20%[**+8.33**]

Table 9: Perplexity and zero-shot evaluation for the quantization on LLaMA-2 models. FP16 denotes the original model precision, and W8A8 denotes the model quantized to INT8 for both weights and activations.