IntactKV: Improving Large Language Model Quantization by Keeping Pivot Tokens Intact

Anonymous ACL submission

Abstract

Large language models (LLMs) excel in natural language processing but demand intensive computation. To mitigate this, various quantization methods have been explored, yet they compromise LLM performance. This paper unveils a previously overlooked type of outlier in LLMs. Such outliers are found to allocate most of the attention scores on initial tokens of input, termed as *pivot tokens*, which is crucial to the performance of quantized LLMs. Given that, we propose *IntactKV* to generate the KV cache of pivot tokens losslessly from the full-precision model. The approach is simple and easy to combine with existing quantization solutions. Besides, IntactKV can be calibrated as additional LLM parameters to boost the quantized LLMs further. Mathematical analysis also proves that IntactKV effectively reduces the upper bound of quantization error. Empirical results show that IntactKV brings consistent improvement and achieves lossless weight-only INT4 quantization on various downstream tasks, leading to the new state-of-the-art for LLM quantization.

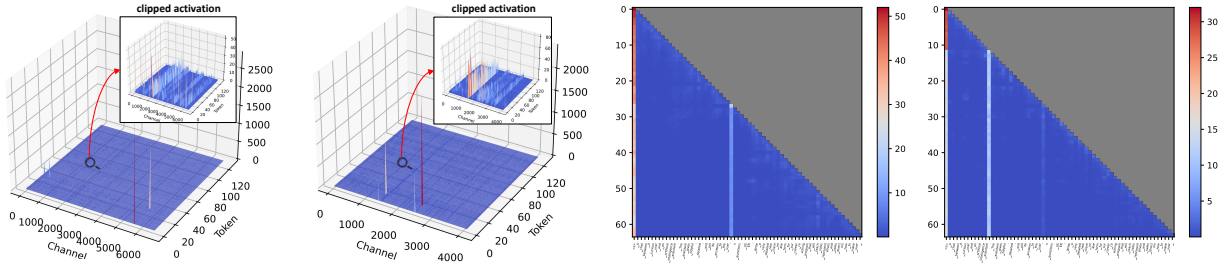
1 Introduction

Large language models (LLMs) have achieved remarkable progress in various tasks and benchmarks in natural language processing (Brown et al., 2020; Bubeck et al., 2023; Touvron et al., 2023a; Team et al., 2023). Nonetheless, the rise of LLMs also increases computational intensity and memory requirements. This motivates various research to decrease the inference cost of LLMs, e.g., quantizaiton (Frantar et al., 2022; Shao et al., 2024; Lin et al., 2023), pruning (Frantar and Alistarh, 2023; Liu et al., 2023b; Sun et al., 2023), and speculative decoding (Chen et al., 2023; Leviathan et al., 2023; Cai et al., 2024), e.t.c.

Among these methods, network quantization converts the network parameters or activations from floating-point to fixed-point formats, which is a popular technique to reduce the model size and computational resources. Nevertheless, quantization inevitably affects the performance of LLMs. The leading cause comes from the outliers in LLM activations, which are sensitive to network quantization (Dettmers et al., 2022; Xiao et al., 2023; Lin et al., 2023). As workarounds, there are efforts to either use mixed-precision formats (Dettmers et al., 2022) or re-scale network weights of the outlier channels (Lin et al., 2023). These methods are all built based on the premise that outliers persist in fixed channels across all tokens. However, we find this is not the case for all outliers in LLMs.

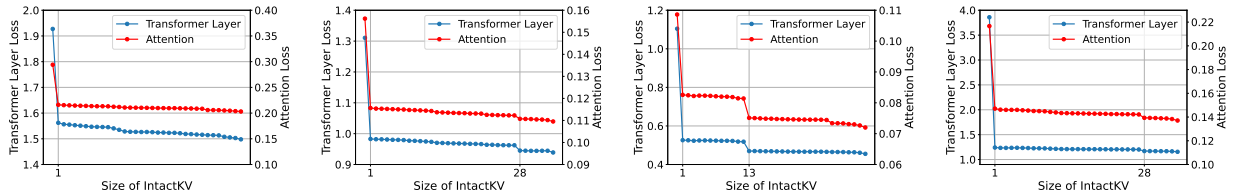
In this paper, we discover a new type of outlier that is overlooked by previous quantization methods. These outliers exhibit extremely high values at only the `[BOS]` and some other common tokens (e.g., "," and ".") at the beginning of the input, which is referred to as ***pivot tokens***. We find the extreme values of these outliers make the self-attention concentrate on the pivot tokens, leaving the rest of the tokens untouched. This is also known as attention sinks (Xiao et al., 2024), which is critical to the model performance (Xiao et al., 2024; Bondarenko et al., 2023). The effect of quantization on these pivot tokens should be carefully studied to improve the quantized LLMs.

Towards that end, we are motivated to propose IntactKV, a simple strategy that is orthogonal to most existing quantization solutions. The key idea behind IntactKV is to *generate the KV cache of pivot tokens from the full-precision model*. By keeping the KV cache of pivot tokens intact, quantization error accumulated on the output of self-attention will be effectively alleviated in the rest of the decoding steps. Moreover, IntactKV can also serve as *extra trainable parameters* in addition to the LLM backbone. The calibration process of IntactKV follows the convention of PTQ (Bai et al., 2022; Frantar et al., 2022; Lin et al., 2023), which further decreases the quantization error. To



(a) Output activations of LLaMA-30B Layer 24 (b) Output activations of LLaMA-2-7B Layer 24 (c) Attention map of LLaMA-30B Layer 24 (d) Attention map of LLaMA-2-7B Layer 24

Figure 1: Visualizations of Transformer output and attention scores of LLaMA-30B and LLaMA-2-7B. Observations: (1) There are token-specific outliers that can be orders of magnitudes larger than the rest of the tokens (enlarged in the box). Such tokens occur at the [BOS] token, the 28th token "'" in LLaMA-30B and 13th token "." in LLaMA-2-7B, which are referred to as ***pivot tokens***; (2) These outliers over pivot tokens make the attention scores concentrated on themselves, which are likely to be affected by quantization. More details can be found in Appendix C.1.



(a) LLaMA-13B (b) LLaMA-30B (c) LLaMA-2-7B (d) LLaMA-2-70B

Figure 2: The mean squared error (MSE) of the last Transformer layer and attention layers w.r.t. the varying sizes of IntactKV. Observations: (1) The MSE continues to drop as the size of IntactKV increases. (2) Including the pivot tokens' KV cache in IntactKV leads to the most significant decrease in the quantization loss, demonstrating the importance of the pivot tokens' KV cache. More experiment details can be found in Appendix D.

get more insights from IntactKV, we also provide mathematical analysis and the results show that IntactKV can effectively lower the upper bound of quantization error.

Empirical results show that IntactKV consistently improves the capability of quantized models on various open-sourced LLMs (e.g., LLaMA and Vicuna) and different downstream tasks such as C4, MMLU, commonsense QA, and MT-bench. When armed with AWQ (Lin et al., 2023), IntactKV achieves new state-of-the-art quantization results, e.g., lossless INT4 weight-only quantization for Vicuna-v1.5 on commonsense QA tasks. Moreover, fine-tuning IntactKV with INT4 quantization even matches the full-precision model on aligning with human preference, as evaluated by GPT-4 (Bubeck et al., 2023) on MT-bench.

2 Motivation

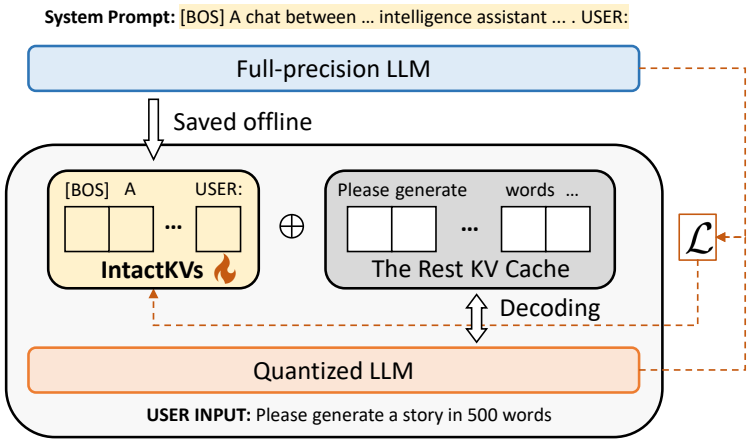
2.1 Preliminaries on LLM Quantization

Network quantization is popularly studied in the literature of efficient LLMs (Frantar et al., 2022; Lin et al., 2023; Shao et al., 2024). It allows larger throughput by reducing the model size and leads to practical inference speedup. Given the full-precision weight $\mathbf{w}$, quantization aims to convert it to the low-bit representation $\hat{\mathbf{w}}$. The general b-bit uniform quantization $\mathcal{Q}_b(\cdot)$ can be represented as

$$\hat{\mathbf{w}} = \mathcal{Q}_b(\mathbf{w}) = s \cdot \Pi_{\Omega(b)}(\mathbf{w}/s), \qquad (1)$$

where s is the quantization step size, and $\Pi_{\Omega(b)}$ is the projection function onto the set of b-bit integers $\Omega(b) = \{0, 1, ..., 2^b - 1\}$. While we mainly focus on weight-only quantization, Equation 1 can be similarly used to quantize activations and KV cache of LLMs to increase the inference throughput (Xiao et al., 2023; Shao et al., 2024; Hooper et al., 2024).

Following most existing works in LLM quantization, we focus on post-training quantization (PTQ) (Frantar et al., 2022; Lin et al., 2023), since it does not introduce extra training overhead as those in quantization-aware training (QAT) (Liu et al., 2023a; Li et al., 2024). Quantization inevitably downgrades LLMs in low-bit settings, where the outliers in quantized LLMs are found to be the cause of the deterioration (Dettmers et al., 2022). In the next, we study the details of how these outliers affect the LLM quantization.



(a) The overview of IntactKV.

Algorithm: Pseudo code for Inference with IntactKV

```
def inference(fp16_model, quantized_model,
              queries, prompt):
    # get IntactKV
    intactkv = fp16_model(
        input=prompt
    ).past_key_values

    # discard the full-precision model
    del fp16_model

    # inference with IntactKV
    outputs = []
    for query in queries:
        output = quantized_model(
            input=query[len(prompt):],
            past_key_values=intactkv
        )
        outputs.append(output)

    return outputs
```

(b) Pseudo Code of Inference.

Figure 3: The overview of the proposed IntactKV applied for the supervised fine-tuned LLM. The full-precision model takes the system prompt as input and generates the IntactKV losslessly as the prefix concatenated with the rest of the KV cache of quantized LLMs. IntactKV can be further calibrated by minimizing the mean square error $\mathcal{L}$ between the full-precision and quantized LLM.

2.2 Revisiting Outliers in LLMs

We discover a new type of outlier that is specific to particular tokens, which leads the attention sink (Xiao et al., 2024) that is critical to the performance of LLMs.

A New Variant of Outlier. Different from the outliers that persist in fixed channels across different tokens (Dettmers et al., 2022; Xiao et al., 2023; Lin et al., 2023), we find a new variant of outlier that is only specific to some initial tokens of the input sequence. By visualizing the activation of Transformer layer output in Figure 1a and Figure 1b, there exist peaks with magnitudes over 1e3. These outliers can be hundreds of times larger than the previous outliers that persist in fixed channels across all tokens, as enlarged in Figure 1a and Figure 1b. It is found that such huge outliers usually occur at the `[BOS]` token and some other uninformative tokens (e.g., "." or ",") at particular channels, regardless of the rest of the input sequence. More visualizations can be found in Appendix C. We thus name these tokens ***pivot tokens*** given their dominating values in the activation.

Pivot Tokens Exhibit Attention Sinks. We hypothesize that the outliers over these pivot tokens may propagate to queries and keys in the self-attention. Consequently, the attention scores will be concentrated on these pivot tokens than the rest ones, a.k.a *attention sinks* (Xiao et al., 2024). To verify the hypothesis, we plot the attention scores in Figure 1c and Figure 1d. It can be found that the pivot tokens indeed dominate the attention scores, especially for the first token (i.e., `[BOS]`). This corresponds to the observations in attention sinks (Xiao et al., 2024), which are empirically verified to be critical to the model performance. The recent study by (Bondarenko et al., 2023) also shows that concentrating on these tokens naturally helps the attention head do nothing but simply a partial update of the residual. In the decoding stage of LLMs, all generated tokens need to interact with pivot tokens through self-attention. However, as mentioned in Section 2.1, network quantization would inevitably distort the output from the full-precision model. The concentrated scores of pivot tokens thus can be further deviated by quantization, which downgrades the model performance.

3 Method

In this section, we introduce IntactKV, a simple and easy-to-implement method to improve the quantized LLMs. The key idea behind this is to keep the KV cache of the pivot tokens intact, i.e., without any distortion raised by quantization. An overview of our method can be found in Figure 3.

3.1 Preserving the KV Cache of Pivot Tokens

According to Section 2.2, the attention sinks of pivot tokens are likely to deteriorate by quantization. To alleviate this issue, we propose IntactKV, a simple and effective strategy to keep these pivot tokens intact. Specifically, as illustrated in Figure 3a, we leverage the full-precision LLM to

generate the lossless KV cache of pivot tokens, which is saved offline. The quantized LLM then loads IntactKV as the prefix to concatenate with the rest of the KV cache and continues with the regular auto-regressive decoding. The pseudo code of the inference scheme with IntactKV is presented in Figure 3b.

In order to study the benefits of IntactKV, we conduct a preliminary test on the mean square error (MSE) of the attention and transformer layer output. From Figure 2, it is natural that the increasing size of IntactKV gives the monotonically decreasing MSE on both the attention and transformer layers. More importantly, it is found the same tokens in Section 2.2 (e.g., `[BOS]` and other delimiter tokens) give the most significant decrease on the MSE, which demonstrates the importance of their KV cache. This aligns with the observations in Figure 1 that pivot tokens exhibit outliers with extreme values and attention sinks.

The Choice of Pivot Tokens and IntactKV. It is the key design to choose the pivot tokens and the associated IntactKV. Given the observations in Figure 2, one can naively pick pivot tokens with the most MSE reduction for IntactKV. However, this is in fact not the case. Since IntactKV acts as the prefix to the KV cache of quantized LLMs, it must start from the very first token, and be consecutive in length. This ensures it to be input agnostic, and the full-precision LLMs can be safely discarded once IntactKV is generated. Next, we provide practical solutions to this problem for different LLMs.

- For pre-trained LLMs, we propose the IntactKV of size one that only contains `[BOS]` KV cache. It is a convention to prepend `[BOS]` to the input of pre-trained LLMs. Moreover, as illustrated in Section 2, `[BOS]` is the pivot token with extreme outlier and attention scores. Besides, the KV cache of `[BOS]` has a great impact on the MSE of the quantized model. Employing a lossless `[BOS]` KV cache is thus believed to decrease the quantization loss.

- For supervised fine-tuned (SFT) models, when the input follows the system prompt, we argue that extending IntactKV to the same length of the system prompt can further improve quantized LLMs. In addition to `[BOS]`, other tokens appearing at the beginning of the input sequence also have the potential to serve as pivot tokens (see Figure 1). The system prompt is usually prepended to the input, which allows it to cover more pivot tokens. As shown in Figure 2, remedying the quantization error of these pivot tokens' KV cache can be helpful to compensate for the quantization error. We find that for Vicuna models, system prompt is enough to cover all the pivot tokens, more details can be found in Appendix C.3.

Overhead of IntactKV. Finally, we highlight that IntactKV does not introduce extra latency overhead during inference. Besides, as IntactKV is pre-computed, the pre-filling stage of the quantized LLMs can be accelerated as well. The memory overhead to save IntactKV is also negligible compared with the LLM backbone. For instance, there are only 34 tokens of the system prompt for Vicuna-v1.5-7B, and thus IntactKV takes only 0.13% of the LLM model parameters.

3.2 IntactKV as Trainable Parameters

Since IntactKV is pre-computed and saved offline, it can be treated as extra trainable parameters aside from the LLM backbone to further boost the quantized LLMs. Despite there being no information loss at the pivot tokens, the quantization may still introduce errors to the KV cache during the decoding stage. As shown in Figure 3a, we calibrate IntactKV to compensate for the quantization error accumulated in the following tokens. While there are various metrics to characterize the quantization discrepancy (Frantar et al., 2022; Shao et al., 2024; Liu et al., 2023a), we adopt the mean square error of the transformer layer output between the full-precision LLM and quantized LLM, a simple yet most widely used metric, i.e.,

$$\mathcal{L}(\Theta) = \frac{1}{2}\sum_{l=1}^{L} \| f_l(\mathbf{w}, \mathbf{x}) - f_l(\hat{\mathbf{w}}, \mathbf{x}; \Theta) \|_2^2, \quad (2)$$

where Θ denotes the set of IntactKV, f_l is the mapping function for the l-th Transformer layer, and L is the number of Transformer layers in LLM. $\mathbf{x}$ is the input sequence, while $\mathbf{w}, \hat{\mathbf{w}}$ are full-precision and quantized weights respectively. Note that the full-precision model is only required during the calibration process, and it can be then discarded afterward. It is empirically found that calibration of system prompt IntactKV in SFT models generally gives more improvement than the calibration of `[BOS]` token in pre-trained LLMs. This matches the intuition that a larger size of In-

TACTKV increases the potential to compensate for quantization errors.

As we focus on the post-training quantization, the training of INTACTKV is highly lightweight since the only learnable parameters introduced are INTACTKV, i.e., the KV cache of pivot tokens. It takes only as few as 20 epochs on a calibration set with 128 samples. Besides, training with a quantized model further lowers the memory cost.

3.3 Theoretical Analysis

In this section, we provide a theoretical view of how the proposed INTACTKV benefits the quantized LLM. For the clarity of presentation, our analysis is built on the self-attention module of a Transformer layer, while it can be readily extended to the FFN module and multiple layers.

Specifically, we denote $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n\times d}$ as the KV cache during the decoding stage, and $\boldsymbol{q} \in \mathbb{R}^d$ is the query vector, where n and d are the sequence length and head dimension. Recall that the output of each attention head $\boldsymbol{h} \in \mathbb{R}^d$ is computed as

$$\boldsymbol{h} = \text{softmax}(\boldsymbol{q}\boldsymbol{K}^\top/\sqrt{d})\boldsymbol{V}\boldsymbol{W}^O, \qquad (3)$$

where $\boldsymbol{W}^O \in \mathbb{R}^{d\times d}$ is the weight matrix of the projection layer. By quantizing the LLMs, there will be errors accumulated on the KV cache, denoted as $\Delta\boldsymbol{K}, \Delta\boldsymbol{V} \in \mathbb{R}^{n\times d}$. Therefore, we are interested in showing how $\Delta\boldsymbol{K}$ and $\Delta\boldsymbol{V}$ are propagated to the change of attention head $\Delta\boldsymbol{h}$, and to what extent INTACTKV alleviates the distortion.

Theorem 1. *Given the query vector $\boldsymbol{q} \in \mathbb{R}^d$ and the change of KV caches $\Delta\boldsymbol{K}, \Delta\boldsymbol{V} \in \mathbb{R}^{n\times d}$, the change of the attention head $\Delta\boldsymbol{h}$ is bounded by*

$$\begin{aligned}\|\Delta\boldsymbol{h}\|_2 \le\, & C_1\|\Delta\boldsymbol{K}\|_{2,\infty}\|\Delta\boldsymbol{V}\|_F + \\ & + C_2\|\Delta\boldsymbol{K}\|_{2,\infty} + C_3\|\Delta\boldsymbol{V}\|_F,\end{aligned}$$

where $C_1 = \frac{n^{3/2}}{\sqrt{d}}C_3\|\boldsymbol{q}\|_2, C_2 = C_1\|\boldsymbol{V}\|_2$ and $C_3 = \|\boldsymbol{W}^O\|_2$.

The proof to Theorem 1 can be found in Appendix A. We preserve the terms w.r.t. $\Delta\boldsymbol{K}$ and $\Delta\boldsymbol{V}$ of interests, and leave the rest as constants. Note that $\Delta\boldsymbol{K}$ can be further separated by the pivot tokens $\Delta\boldsymbol{K}_p$ and rest tokens $\Delta\boldsymbol{K}_{\setminus p}$, and similar notations hold for $\Delta\boldsymbol{V}$. Therefore, we have $\|\Delta\boldsymbol{K}\|_{2,\infty} = \max\left(\|\Delta\boldsymbol{K}_p\|_{2,\infty}, \|\Delta\boldsymbol{K}_{\setminus p}\|_{2,\infty}\right)$, and $\|\Delta\boldsymbol{V}\|_F = \sqrt{\|\Delta\boldsymbol{V}_p\|_F^2 + \|\Delta\boldsymbol{V}_{\setminus p}\|_F^2}$. With INTACTKV we have $\|\Delta\boldsymbol{K}_p\|_{2,\infty} = \|\Delta\boldsymbol{V}_p\|_F = 0$ since they are generated losslessly, which decreases the upper bound of $\|\Delta\boldsymbol{h}\|_2$. Moreover, it can further reduce the bound by incorporating more pivot tokens. This also aligns with the observation in Figure 2 that a larger size of INTACTKV gives a lower MSE of the attention module.

4 Experiments

4.1 Settings

Models. We evaluate the proposed INTACTKV on various sizes of open-sourced LLMs, including LLaMA (Touvron et al., 2023a) (7B-65B), LLaMA-2 (Touvron et al., 2023b) (7B-70B), Vicuna-v1.3 (7B-33B) and Vicuna-v1.5 (Touvron et al., 2023b) (7B-13B). We denote models that keep intact `[BOS]` KV as INTACTKV$_{[B]}$, and models that keep intact system prompt KV as INTACTKV$_{[P]}$.

Quantization Methods. We mainly consider weight-only quantization methods, including round-to-nearest quantization (RTN), GPTQ (Frantar et al., 2022), the state-of-the-art OmniQuant (Shao et al., 2024) and AWQ (Lin et al., 2023). For GPTQ, we use AutoGPTQ[1] with C4 calibration set following GPTQ paper (Frantar et al., 2022) to reproduce all results. For AWQ[2] and OmniQuant[3], we directly load the officially released quantization parameters of LLaMA models for evaluation and reproduce results on Vicuna models with their official code. We use Pile (Gao et al., 2020) calibration set for AWQ and WikiText2 (Merity et al., 2016) calibration set for OmniQuant, following (Lin et al., 2023; Shao et al., 2024). We adopt asymmetric group-wise quantization with a group size of 128 and mainly focus on INT3 and INT4 quantization since INT8 is empirically lossless on various task metrics (Dettmers et al., 2022).

While INTACTKV can be readily combined with these existing quantization methods, we mainly apply INTACTKV for AWQ in the main experiments due to the space limitation and the state-of-the-art performance of AWQ. We shall provide a more comprehensive evaluation of combining INTACTKV with other quantization methods in Section 4.3. Moreover, aside from weight-only quantization, the proposed INTACTKV can be similarly applied for KV cache quantization, as detailed in Section 4.4. Nonetheless, it is non-trivial to apply INTACTKV for activation quantization, since the quantization over KV cache cannot keep them

[1] https://github.com/AutoGPTQ/AutoGPTQ
[2] https://github.com/mit-han-lab/llm-awq
[3] https://github.com/OpenGVLab/OmniQuant

Method	LLaMA-7B	LLaMA-13B	LLaMA-30B	LLaMA-65B	LLaMA-2-7B	LLaMA-2-13B	LLaMA-2-70B
FP16	7.36	6.82	6.15	5.83	7.11	6.58	5.59
RTN	9.15	7.89	6.85	6.33	8.79	7.43	6.12
GPTQ	8.59	7.49	6.73	6.29	9.41	7.25	6.27
OmniQuant	8.24	7.38	6.64	6.17	8.21	7.23	5.96
AWQ	8.26	7.38	6.59	6.16	8.12	7.15	5.91
+IntactKV$_{[B]}$	**8.12**	**7.36**	**6.54**	**6.12**	**8.00**	**7.12**	**5.89**

Table 1: INT3-group128 quantization results of LLaMA and LLaMA-2 Models on C4 dataset.

Model	Method	MMLU (0 shot)					MMLU (5 shot)				
		Hums	STEM	Social	Others	Avg.	Hums	STEM	Social	Others	Avg.
Vicuna-v1.5-13B	FP16	50.48%	43.70%	62.72%	62.74%	54.54%	51.97%	44.96%	65.26%	62.40%	55.78%
	RTN	46.61%	41.32%	58.92%	57.53%	50.69%	47.14%	42.81%	59.38%	58.17%	51.44%
	GPTQ	48.35%	40.99%	59.25%	57.99%	51.38%	49.63%	43.04%	60.22%	60.09%	52.95%
	OmniQuant	**49.52%**	41.22%	59.47%	57.74%	51.82%	49.12%	**44.40%**	60.48%	58.95%	52.86%
	AWQ	48.82%	41.72%	61.03%	58.30%	52.16%	49.52%	43.01%	61.72%	58.73%	52.92%
	+IntactKV$_{[B]}$	49.31%	**42.18%**	**61.20%**	**59.28%**	**52.68%**	**50.31%**	43.37%	**61.91%**	**59.93%**	**53.58%**

Table 2: INT3-group128 quantization results of Vicuna-v1.5-13B on 0-shot and 5-shot MMLU benchmarks.

Task Acc	MMLU (5 shot) average					Common Sense QA (0 shot) average				
Vicuna Family	v1.5-7B	v1.5-13B	v1.3-7B	v1.3-13B	v1.3-33B	v1.5-7B	v1.5-13B	v1.3-7B	v1.3-13B	v1.3-33B
FP16	49.84%	55.78%	47.12%	52.10%	59.30%	65.33%	68.38%	64.52%	67.22%	69.53%
RTN	44.62%	51.44%	39.33%	44.56%	53.18%	60.99%	65.40%	59.13%	63.23%	67.19%
GPTQ	43.99%	52.95%	40.12%	47.83%	55.84%	58.09%	65.78%	59.80%	64.03%	66.68%
OmniQuant	46.54%	52.86%	43.18%	47.92%	55.12%	61.73%	65.23%	61.08%	64.81%	67.58%
AWQ	46.45%	52.92%	43.08%	48.56%	56.09%	61.86%	66.01%	60.94%	64.53%	67.67%
+IntactKV$_{[B]}$	**46.87%**	**53.58%**	**44.67%**	**49.05%**	**56.91%**	**62.49%**	**66.93%**	**61.93%**	**65.02%**	**67.90%**

Table 3: INT3-group128 quantization results of various Vicuna models on 5-shot MMLU and 0-shot QA tasks.

intact. We leave it as future work and more discussions are provided in Section 6.

Evaluation. For pre-trained LLMs (i.e., LLaMA and LLaMA-2), we report the perplexity (PPL) of language generation on C4 (Raffel et al., 2020) and WikiText2 (Merity et al., 2016) dataset. For SFT models (i.e., Vicuna-v1.3 and v1.5), we conduct evaluation over a wide range of downstream tasks. We test the zero and five-shot performance on the Massively Multitask Language Understanding (MMLU) (Hendrycks et al., 2020) benchmark. Meanwhile, we also evaluate seven zero-shot commonsense QA tasks: OBQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), ARC-Challenge, ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), and LAMBADA (Paperno et al., 2016). Additionally, we evaluate quantized Vicuna on MT-bench (Zheng et al., 2023), a high-quality dataset consisting of 80 open-ended multi-turn questions, to gauge their alignment with human preferences. The responses generated by quantized models are judged by GPT-4 with a total score of 10. More evaluation details can be found in Appendix E.

Implementation Details For evaluation on PPL, MMLU, and commonsense QA tasks, we adopt IntactKV$_{[B]}$ that only includes `[BOS]` KV since the input sequence of these tasks does not use any system prompt. For evaluation of SFT models on MT-bench, we adopt IntactKV$_{[P]}$ to keep an intact system prompt KV cache. The system prompt of Vicuna can be found in Appendix B. For training the cached IntactKV, we randomly sample 128 samples from ShareGPT[4] dataset as our calibration dataset, consisting of multi-turn ChatGPT (OpenAI, 2022) conversations. The layer-wise MSE defined in Equation 2 is calculated on the response of ChatGPT. We use AdamW optimizer with learning rate 2×10^{-4}, training for 160 optimizer update steps with a gradient accumulation step of 16, i.e., 20 epochs. As mentioned in Section 3.2, training IntactKV$_{[B]}$ leads to comparable performance compared with vanilla IntactKV. Instead, the calibration of IntactKV$_{[P]}$ has more potential to improve quantized LLMs with longer system prompt. Thus, we primarily evaluate the trainable

[4] https://huggingface.co/datasets/Aeala/ShareGPT_Vicuna_unfiltered

Model	#bits	Method	OBQA	WinoGrande	ARC-C	ARC-E	BoolQ	HellaSwag	LAMBADA	Avg
Vicuna-v1.5-13B	FP16	-	45.40%	71.51%	50.68%	74.87%	85.29%	77.50%	73.43%	68.38%
	w3g128	RTN	42.00%	70.01%	47.44%	**72.77%**	82.20%	74.18%	69.20%	65.40%
		GPTQ	42.40%	69.53%	48.46%	71.84%	83.76%	74.48%	70.00%	65.78%
		OmniQuant	43.40%	68.51%	47.53%	71.09%	82.32%	73.92%	69.84%	65.23%
		AWQ	44.00%	68.75%	48.21%	71.76%	83.58%	75.09%	70.68%	66.01%
		+IntactKV$_{[B]}$	**45.40%**	**70.32%**	**48.38%**	72.14%	**85.20%**	**75.23%**	**71.86%**	**66.93%**
	w4g128	RTN	44.20%	70.80%	48.98%	73.82%	84.68%	76.36%	73.04%	67.41%
		GPTQ	**45.80%**	70.96%	**50.51%**	73.99%	85.47%	76.70%	**73.43%**	68.12%
		OmniQuant	43.80%	70.24%	49.74%	73.61%	84.59%	76.35%	72.54%	67.27%
		AWQ	44.00%	72.06%	49.15%	73.44%	85.17%	77.00%	72.77%	67.66%
		+IntactKV$_{[B]}$	45.40%	**73.09%**	49.57%	**74.45%**	**85.66%**	**77.32%**	72.75%	**68.32%**

Table 4: Quantization results of Vicuna-v1.5-13B on seven 0-shot commonsense QA tasks.

#bits	Method	Vicuna-v1.5-7B	Vicuna-v1.5-13B
FP16	-	5.31	5.52
w3g128	RTN	4.34	5.13
	OmniQuant	4.78	5.05
	AWQ	4.74	5.17
	+IntactKV$_{[P]}$	4.68	5.34
	+IntactKV$_{[P]}$+FT	**4.84**	**5.44**
w4g128	RTN	5.18	5.47
	OmniQuant	5.09	5.48
	AWQ	5.22	5.28
	+IntactKV$_{[P]}$	5.32	5.35
	+IntactKV$_{[P]}$+FT	**5.36**	**5.50**

Table 5: GPT-4 evaluation of quantized Vicuna-v1.5 models on MT-Bench. The scores are on a scale of 10.

#bits	Method	Vicuna-v1.5-7B	Vicuna-v1.5-13B
FP16	-	5.31	5.52
w3g128	RTN	4.34	5.13
	+IntactKV$_{[P]}$	4.72	5.27
	+IntactKV$_{[P]}$+FT	**4.73**	**5.30**
	OmniQuant	4.78	5.05
	+IntactKV$_{[P]}$	**4.94**	5.10
	+IntactKV$_{[P]}$+FT	4.85	**5.24**
	AWQ	4.74	5.17
	+IntactKV$_{[P]}$	4.68	5.34
	+IntactKV$_{[P]}$+FT	**4.84**	**5.44**

Table 6: Compatibility of IntactKV with various quantization methods. The scores are on a scale of 10.

IntactKV$_{[P]}$ with system prompt as pivot tokens in the following experiments.

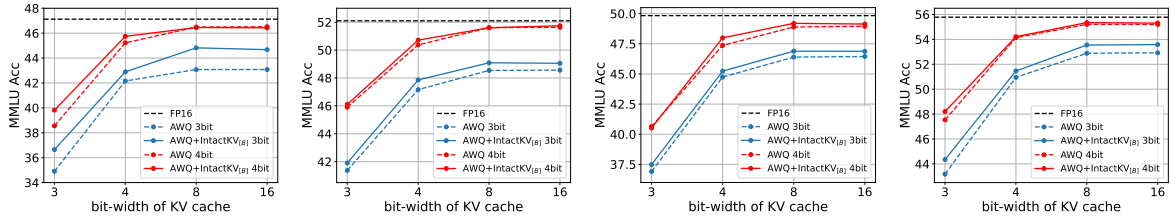
4.2 Main Results

Results on Language Generation Tasks. We first integrate our proposed IntactKV with AWQ on LLaMA and LLaMA-2 models. The effect of this integration on model performance was measured by the perplexity (PPL) metric, with results on the C4 dataset detailed in Table 1, and results on the WikiText2 dataset in Appendix F.1. As indicated in these tables, IntactKV notably enhances the generative capabilities of AWQ-quantized models across various sizes, consistently surpassing the prior state-of-the-art (SOTA) method, OmniQuant. These findings demonstrate the efficacy of IntactKV in improving quantized LLMs and particularly highlight the effectiveness of utilizing the KV cache from full-precision models.

Results on MMLU Tasks. For SFT models, we implement IntactKV on the AWQ-quantized Vicuna models and evaluate the multi-task problem-solving ability on the MMLU benchmark. Table 2 presents the detailed zero-shot and five-shot results for Vicuna-v1.5-13B. The results demonstrate that IntactKV significantly enhances the performance of the AWQ-quantized model across all categories of tasks for Vicuna-v1.5-13B. Moreover, the performance of other model sizes under the five-shot setting is outlined in Table 3. Remarkably, AWQ+IntactKV exhibits superior performance over OmniQuant, achieving an average improvement of 1.09% across five model sizes. More results on MMLU are provided in Appendix F.2.

Results on Commonsense QA Tasks. We further evaluate the quantized Vicuna models on zero-shot commonsense QA tasks. The results of Vicuna-v1.5-13B, as detailed in Table 4, indicate that IntactKV enables significant improvements over AWQ. Notably, our AWQ+IntactKV largely surpasses GPTQ and OmniQuant, especially under low-bit quantization. Additionally, Table 3 presents the average accuracy for various sizes of Vicuna models. In these evaluations, our AWQ+IntactKV consistently achieves the best zero-shot performance, which strongly demonstrates the efficacy of our proposed IntactKV. More results on Commonsense QA tasks can be found in Appendix F.3.

Results on MT-Bench. To evaluate the quantized models' generation capabilities in multi-turn conversations and their alignment with human preferences, we use GPT-4 to score the responses of



(a) Vicuna-v1.3-7B (b) Vicuna-v1.3-13B (c) Vicuna-v1.5-7B (d) Vicuna-v1.5-13B

Figure 4: Results of KV cache quantization with different bit-widths on 5-shot MMLU benchmark. Note that this is additional to INT3/4 weight quantization. Blue and red lines indicate quantizing model weights to INT3 and INT4, respectively. More experiment details can be found in Appendix G.

quantized models on MT-Bench. On MT-bench, we further fine-tune IntactKV, denoted as IntactKV+FT. As shown in Table 5, our IntactKV+FT significantly boosts the quantized model and consistently surpasses the GPTQ and OmniQuant for both INT3-g128 and INT4-g128 settings. For example, the 3-bit Vicuna-v1.5-13B quantized by AWQ has been improved from 5.17 to 5.34 by using the IntactKV, which can be further boosted to 5.44 with further fine-tuning. Remarkably, with trainable IntactKV, the setting of INT4-g128 even matches the full-precision model, while all other methods still lag behind the full-precision model by a considerable margin. These results demonstrate the effectiveness of treating IntactKV as trainable parameters. Notably, the training process for the 7B model takes only 10 minutes on one NVIDIA H800 GPU, which is quite lightweight.

4.3 IntactKV with Other PTQ Methods

To assess IntactKV's compatibility with different quantization methods, we build IntactKV upon various PTQ methods (i.e., RTN, OmniQuant, and AWQ) and evaluate on MT-bench with INT3-g128 quantization. As shown in Table 6, integrating different PTQ methods with IntactKV can lead to an average boost of 0.16 in the final score for Vicuna-v1.5-7B, and 0.12 for Vicuna-v1.5-13B. Fine-tuning IntactKV further improves the performance of quantized models. Our IntactKV+FT consistently surpasses the performance of original quantized models for all three different quantization methods, raising the final score by 0.19 for Vicuna-v1.5-7B and 0.21 for Vicuna-v1.5-13B on average. These findings validate IntactKV's capability of acting as a lightweight plugin, consistently improving various quantized models with negligible extra costs.

4.4 Extension to KV Cache Quantization

IntactKV can be readily applied to KV cache quantization to further decrease memory requirements. Similar to weight-only quantization, the quantized KV cache needs to be de-quantized before the matrix multiplication. Nonetheless, the proposed IntactKV is kept in FP16 since it is more sensitive to quantization. Also, the concatenation between the FP16 IntactKV and the rest de-quantized KV cache incurs negligible extra overhead. From Figure 4, IntactKV notably improves AWQ across different models and KV cache bit widths under the INT3 weight quantization. For INT4 weight quantization, IntactKV still gains an average accuracy increase of 0.27% over AWQ. We also notice that quantizing the KV cache to INT8 leads to almost no performance drop on the MMLU benchmark. When equipped with IntactKV, INT8 KV cache can even surpass vanilla AWQ-quantized models with FP16 KV cache, especially under INT3 weight quantization.

5 Conclusions

In this paper, we propose IntactKV, a simple and easy-to-combine method to improve large language model quantization. The research is motivated by the previously overlooked outliers over pivot tokens, which lead to attention sinks that are critical to the performance of quantized LLMs. By generating IntactKV with the full-precision model, the quantization error accumulated over the attention scores can be effectively alleviated. IntactKV can also be calibrated as additional parameters to the LLM backbone, further improving the quantized LLMs. Experiments show that combining the proposed IntactKV gives consistent improvement on various sizes of LLMs and across multiple downstream tasks, leading to new state-of-the-art results for large language model quantization.

6 Limitations

Firstly, it is non-trivial to integrate INTACTKV into activation quantization. For activation quantization, the whole KV cache needs to be quantized to low bits to exploit integer multiplications in self-attention, which contradicts our idea of keeping pivot tokens' KV cache intact. Since INTACTKV is treated as extra cached parameters, a straightforward solution is to quantize INTACTKV with suitable quantization methods such as GPTQ. We leave this as future work. Secondly, more experiments may be needed for LLM evaluation. LLMs are being applied to a wide range of tasks, posing high demands on various model abilities. When quantizing LLMs to low bits, these abilities may be affected to varying degrees. Therefore, a comprehensive evaluation is required to gauge the capabilities of quantized LLMs. Although we experiment on several downstream tasks, such as PPL, MMLU, commonsense QA, and MT-bench, we note that this may not be enough to assess all abilities of LLMs. For example, how long context affects quantized models still remains unknown.

7 Ethics Statement

The development of LLM quantization techniques can further democratize LLMs, lowering the costs of LLM serving and enabling more people to get access to advanced AI assistants. Nonetheless, LLM itself may inherit certain social biases from training data concerning gender, race, etc. Quantization can not mitigate such biases. Therefore, caution must be taken when using quantized LLMs.

References

Haoli Bai, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, and Michael R Lyu. 2022. Towards efficient post-training quantization of pre-trained language models. *Advances in Neural Information Processing Systems*, 35:1405–1418.

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. 2021. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2023. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2924–2936.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Optq: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2024. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*.

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*.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023a. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023b. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

OpenAI. 2022. Introducing chatgpt.

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*, pages 1525–1534.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The International Conference on Learning Representations*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

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.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

A Proof of Theorem 1

Proof. Denote the output of the softmax function as the score $\boldsymbol{s}$, i.e., $\boldsymbol{s} = \mathrm{softmax}(\frac{\boldsymbol{q}\boldsymbol{K}^\top}{\sqrt{d}})$, and also define the error output from the softmax function as $\Delta\boldsymbol{s}$. To show the error of the attention head, we first justify how the error propagates from the score to the attention head.

$$\begin{aligned}\|\Delta\boldsymbol{h}\|_2 &= \left\|[(\boldsymbol{s}+\Delta\boldsymbol{s})(\boldsymbol{V}+\Delta\boldsymbol{V}) - \boldsymbol{s}\boldsymbol{V}]\boldsymbol{W}^O\right\|_2 \\ &\le (\|\Delta\boldsymbol{s}\|_2\|\boldsymbol{V}+\Delta\boldsymbol{V}\|_2 + \|\boldsymbol{s}\|_2\|\Delta\boldsymbol{V}\|_2)\,\|\boldsymbol{W}^O\|_2 \\ &\le \big(\|\Delta\boldsymbol{s}\|_2(\|\boldsymbol{V}\|_2+\|\Delta\boldsymbol{V}\|_F) + \|\Delta\boldsymbol{V}\|_F\big)\|\boldsymbol{W}^O\|_2,\end{aligned}$$

where the inequalities are because

$$\|\boldsymbol{x}+\boldsymbol{y}\|_2 \le \|\boldsymbol{x}\|_2+\|\boldsymbol{y}\|_2,\ \ \|\boldsymbol{s}\boldsymbol{V}\|_2 \le \|\boldsymbol{s}\|_2\|\boldsymbol{V}\|_2,$$

and $\|\boldsymbol{s}\|_2 \le \|\boldsymbol{s}\|_1 = 1, \|\boldsymbol{V}\|_2 \le \|\boldsymbol{V}\|_F$.

Next, we characterize the error of score $\|\Delta\boldsymbol{s}\|_2$. This is not easy as the error propagates through the softmax function. To proceed, we need the relative condition number of the softmax function. As indicated in (Blanchard et al., 2021),

$$\frac{\|\mathrm{softmax}(\boldsymbol{x}+\Delta\boldsymbol{x})-\mathrm{softmax}(\boldsymbol{x})\|_\infty}{\|\mathrm{softmax}(\boldsymbol{x})\|_\infty} \le \kappa(\boldsymbol{x})\frac{\|\Delta\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty},$$

where $\kappa(\boldsymbol{x}) = n\|\boldsymbol{x}\|_\infty$ ($\boldsymbol{x} \in \mathbb{R}^n$) is an upper bound of the relative condition number of the softmax function. Let $\boldsymbol{x} = \boldsymbol{q}\boldsymbol{K}^\top/\sqrt{d}$ and $\Delta\boldsymbol{x} = \boldsymbol{q}\Delta\boldsymbol{K}^\top/\sqrt{d}$, we have

$$\frac{\|\Delta\boldsymbol{s}\|_\infty}{\|\boldsymbol{s}\|_\infty} \le n\|\Delta\boldsymbol{x}\|_\infty \le \frac{n}{\sqrt{d}}\|\boldsymbol{q}\|_2\|\Delta\boldsymbol{K}\|_{2,\infty}.$$

Considering that the output of the softmax function is a probability, we have $\|\boldsymbol{s}\|_\infty \le 1$. Therefore, we obtain

$$\|\Delta\boldsymbol{s}\|_2 \le \sqrt{n}\|\Delta\boldsymbol{s}\|_\infty \le \frac{n^{2/3}}{\sqrt{d}}\|\boldsymbol{q}\|_2\|\Delta\boldsymbol{K}\|_{2,\infty}.$$

Combining the above ingredients, we derive the main results of the Theorem 1.

□

B System Prompt of Vicuna Models

```
[BOS] A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. USER:
```

Figure 5: System Prompt of Vicuna Models.

C Visualization of Activations and Attention Map

C.1 Implementation Details

We use ShareGPT dataset for our visualizations, where each sample starts with Vicuna system prompt of length 34. We use a randomly sampled sequence of length 128 to visualize the output activations and plot the corresponding attention map of the first 64 tokens. The attention score is mean pooled over different heads.

C.2 Visualization of LLaMA Models

We provide more visualizations of the output activations and attention map of LLaMA models in Figure. 6–12. Similar to our observations in Section 2, we find that pivot tokens only appear at the very beginning of the input sequence, and `[BOS]` always serves as a pivot token.

C.3 Visualization of Vicuna Models

We provide more visualizations of the output activations and attention map of Vicuna models in Figure. 13–17. Although Vicuna models demonstrate stronger performance than LLaMA models of the same size, we are surprised to find that the position of pivot tokens remains unchanged for Vicuna and LLaMA models of the same size. Besides, as shown in Figure. 13–17, we find that the Vicuna system prompt is enough to cover all the pivot tokens in all Vicuna models.

D Experiment Details of Figure 2

We plot the quantization loss of the last Transformer layer as well as the total quantization loss of all attention layers with respect to the size of IntactKV on four different models, i.e., LLaMA-13B, LLaMA-30B, LLaMA-2-7B, and LLaMA-2-70B, covering different model types and model sizes. We use lossless IntactKV generated by the full-precision model to quantify the effect of IntactKV on the quantized model. IntactKV of size s can ensure that the KV cache of the first s tokens of the input sequence is generated by the full-precision model and thus lossless. Quantization loss is computed with MSE loss between the output activations of the quantized model and the full-precision model. We sample 128 sequences from the ShareGPT dataset to construct the validation set, each with a common prompt prefix of length 34. MSE loss is calculated on the tokens after the common prompt prefix. We quantize

Method	LLaMA-7B	LLaMA-13B	LLaMA-30B	LLaMA-65B	LLaMA-2-7B	LLaMA-2-13B	LLaMA-2-70B
FP16	5.69	5.08	4.09	3.52	5.12	4.57	3.12
RTN	6.98	5.88	4.84	4.22	6.21	5.17	3.74
GPTQ	6.62	5.68	4.75	4.20	6.68	5.14	3.79
OmniQuant	**6.18**	**5.46**	4.57	4.00	5.77	4.96	3.58
AWQ	6.34	5.53	4.60	3.95	5.82	4.97	3.53
+IntactKV$_{[B]}$	6.23	5.49	**4.54**	**3.89**	**5.73**	**4.95**	**3.50**

Table 7: INT3-group128 quantization results of LLaMA and LLaMA-2 Models on WikiText2 dataset.

Model	Method	MMLU (0 shot)					MMLU (5 shot)				
		Hums	STEM	Social	Others	Avg.	Hums	STEM	Social	Others	Avg.
Vicuna-v1.5-7B	FP16	45.40%	38.67%	56.16%	55.92%	48.74%	45.78%	39.50%	58.14%	57.46%	49.84%
	RTN	42.06%	34.16%	50.47%	50.59%	44.17%	40.68%	38.60%	50.31%	50.56%	44.62%
	GPTQ	39.89%	33.00%	48.10%	48.46%	42.19%	40.30%	36.28%	50.76%	50.09%	43.99%
	OmniQuant	42.57%	35.98%	51.64%	53.55%	**45.68%**	42.95%	37.57%	53.59%	53.39%	46.54%
	AWQ	42.08%	35.55%	51.61%	51.54%	44.95%	42.55%	38.93%	53.10%	52.78%	46.45%
	+IntactKV$_{[B]}$	42.42%	35.42%	51.71%	51.57%	45.06%	42.95%	38.60%	54.37%	53.15%	**46.87%**
Vicuna-v1.5-13B	FP16	50.48%	43.70%	62.72%	62.74%	54.54%	51.97%	44.96%	65.26%	62.40%	55.78%
	RTN	46.61%	41.32%	58.92%	57.53%	50.69%	47.14%	42.81%	59.38%	58.17%	51.44%
	GPTQ	48.35%	40.99%	59.25%	57.99%	51.38%	49.63%	43.04%	60.22%	60.09%	52.95%
	OmniQuant	49.52%	41.22%	59.47%	57.74%	51.82%	49.12%	44.40%	60.48%	58.95%	52.86%
	AWQ	48.82%	41.72%	61.03%	58.30%	52.16%	49.52%	43.01%	61.72%	58.73%	52.92%
	+IntactKV$_{[B]}$	49.31%	42.18%	61.20%	59.28%	**52.68%**	50.31%	43.37%	61.91%	59.93%	**53.58%**
Vicuna-v1.3-7B	FP16	44.31%	36.28%	53.23%	53.70%	46.71%	44.23%	38.34%	53.82%	53.15%	47.12%
	RTN	38.09%	31.58%	42.35%	44.32%	39.06%	36.81%	32.77%	43.87%	44.79%	39.33%
	GPTQ	39.09%	32.57%	44.59%	46.73%	40.66%	36.94%	33.90%	45.08%	45.81%	40.12%
	OmniQuant	41.36%	33.96%	47.42%	47.69%	42.56%	41.15%	35.45%	48.20%	48.58%	43.18%
	AWQ	40.49%	32.44%	47.06%	49.57%	42.29%	39.64%	36.22%	48.72%	49.11%	43.08%
	+IntactKV$_{[B]}$	41.76%	32.94%	47.74%	49.72%	**43.01%**	41.93%	36.58%	50.37%	50.77%	**44.67%**
Vicuna-v1.3-13B	FP16	47.89%	39.96%	58.86%	57.34%	50.77%	49.78%	40.46%	60.61%	58.24%	52.10%
	RTN	42.06%	32.87%	47.61%	49.51%	43.02%	42.42%	34.46%	50.34%	51.57%	44.56%
	GPTQ	45.06%	35.88%	52.23%	51.26%	46.09%	45.82%	37.57%	54.83%	53.64%	47.83%
	OmniQuant	43.29%	36.65%	51.64%	53.05%	45.95%	45.29%	37.84%	55.02%	54.38%	47.92%
	AWQ	45.14%	36.18%	52.55%	53.79%	46.84%	46.65%	37.64%	55.54%	54.87%	48.56%
	+IntactKV$_{[B]}$	45.91%	36.65%	53.75%	54.60%	**47.64%**	46.57%	38.40%	56.03%	55.95%	**49.05%**
Vicuna-v1.3-33B	FP16	53.73%	44.14%	67.63%	63.54%	56.98%	57.66%	46.32%	69.32%	64.25%	59.30%
	RTN	49.88%	40.13%	61.33%	58.42%	52.26%	51.26%	42.54%	61.75%	57.71%	53.18%
	GPTQ	51.22%	40.03%	61.85%	59.47%	53.05%	54.05%	44.04%	64.35%	61.35%	55.84%
	OmniQuant	51.14%	42.08%	63.60%	59.84%	53.93%	53.77%	43.80%	63.47%	59.69%	55.12%
	AWQ	51.69%	42.74%	63.41%	61.38%	54.57%	54.56%	44.10%	65.36%	60.67%	56.09%
	+IntactKV$_{[B]}$	52.09%	42.68%	63.70%	62.03%	**54.91%**	55.79%	44.90%	65.62%	61.47%	**56.91%**

Table 8: INT3-group128 quantization results of Vicuna models on 0-shot and 5-shot MMLU benchmarks.

the model weights to 3 bits using round-to-nearest quantization with a group size of 128.

E Evaluation Details

PPL. We evaluate PPL following the new evaluation setting in GPTQ official code[5], except that we substitute the first token of each text segment with `[BOS]` token to evaluate the performance of IntactKV.

MMLU. We evaluate MMLU following the original MMLU implementation[6] for 0-shot and 5-shot tasks. We note that when using Vicuna, it is considered more appropriate to fit the input sequences into the Vicuna system prompt. However, the original MMLU implementation does not use the Vicuna system prompt for Vicuna models. In our experiments on Vicuna models, we find that naively fitting the original MMLU prompt into the Vicuna system prompt will harm the final accuracy. Since prompt engineering is out of scope for this paper, we choose to follow the original evaluation setting that does not use the Vicuna system prompt for MMLU evaluation on Vicuna models.

Common Sense Reasoning Tasks. For the seven zero-shot common sense reasoning tasks, we adopt

[5] https://github.com/ist-daslab/gptq

[6] https://github.com/hendrycks/test/pull/13

Model	Method	MMLU (0 shot)					MMLU (5 shot)				
		Hums	STEM	Social	Others	Avg.	Hums	STEM	Social	Others	Avg.
Vicuna-v1.5-7B	FP16	45.40%	38.67%	56.16%	55.92%	48.74%	45.78%	39.50%	58.14%	57.46%	49.84%
	RTN	44.65%	38.47%	53.95%	54.41%	47.61%	44.87%	39.13%	56.45%	55.34%	48.59%
	GPTQ	44.87%	37.08%	54.44%	53.86%	47.37%	45.44%	38.83%	57.33%	56.14%	49.10%
	OmniQuant	45.61%	38.70%	55.64%	56.23%	**48.78%**	45.21%	38.77%	57.17%	57.37%	**49.25%**
	AWQ	45.08%	37.41%	55.64%	55.31%	48.11%	45.44%	38.97%	56.94%	55.74%	48.95%
	+IntactKV$_{[B]}$	45.25%	37.51%	55.93%	55.58%	48.31%	45.33%	39.60%	57.36%	55.74%	49.14%
Vicuna-v1.5-13B	FP16	50.48%	43.70%	62.72%	62.74%	54.54%	51.97%	44.96%	65.26%	62.40%	55.78%
	RTN	50.01%	43.41%	62.33%	62.00%	54.06%	51.31%	43.14%	63.54%	61.63%	54.61%
	GPTQ	50.20%	42.31%	61.62%	61.41%	53.60%	50.10%	43.97%	62.72%	61.01%	54.07%
	OmniQuant	50.01%	43.70%	62.33%	62.00%	**54.12%**	51.67%	43.87%	63.34%	61.81%	54.89%
	AWQ	50.10%	42.94%	61.68%	61.66%	53.77%	52.31%	44.43%	63.18%	61.84%	55.20%
	+IntactKV$_{[B]}$	50.14%	42.84%	61.78%	61.91%	53.84%	52.31%	44.37%	63.67%	61.91%	**55.31%**
Vicuna-v1.3-7B	FP16	44.31%	36.28%	53.23%	53.70%	46.71%	44.23%	38.34%	53.82%	53.15%	47.12%
	RTN	42.78%	36.55%	51.74%	51.48%	45.41%	42.23%	37.08%	52.10%	51.94%	45.53%
	GPTQ	43.40%	34.46%	52.06%	53.45%	45.70%	43.78%	36.41%	53.49%	52.41%	46.32%
	OmniQuant	42.81%	34.72%	52.26%	52.00%	45.26%	43.21%	37.81%	52.62%	53.24%	46.43%
	AWQ	43.53%	36.22%	53.01%	52.53%	46.11%	43.36%	37.74%	53.46%	52.68%	**46.52%**
	+IntactKV$_{[B]}$	43.57%	36.51%	52.29%	53.27%	**46.20%**	43.51%	37.44%	53.17%	52.62%	46.43%
Vicuna-v1.3-13B	FP16	47.89%	39.96%	58.86%	57.34%	50.77%	49.78%	40.46%	60.61%	58.24%	52.10%
	RTN	47.16%	39.00%	56.52%	56.63%	49.64%	49.25%	39.63%	57.85%	57.74%	51.03%
	GPTQ	46.95%	39.30%	57.39%	56.23%	49.74%	49.05%	39.46%	59.02%	57.65%	51.16%
	OmniQuant	47.93%	39.50%	57.98%	57.31%	**50.48%**	49.18%	39.86%	59.41%	58.14%	51.49%
	AWQ	48.03%	39.43%	56.94%	56.76%	50.15%	49.44%	40.49%	59.57%	57.65%	51.63%
	+IntactKV$_{[B]}$	47.91%	39.60%	57.69%	56.79%	50.31%	49.54%	40.23%	60.12%	57.71%	**51.74%**
Vicuna-v1.3-33B	FP16	53.73%	44.14%	67.63%	63.54%	56.98%	57.66%	46.32%	69.32%	64.25%	59.30%
	RTN	53.18%	44.27%	66.88%	62.95%	56.52%	56.73%	45.73%	68.09%	62.49%	58.18%
	GPTQ	52.92%	44.90%	67.05%	63.66%	56.77%	57.13%	45.96%	67.63%	63.11%	58.41%
	OmniQuant	53.22%	44.73%	67.53%	63.05%	56.80%	56.68%	45.46%	68.67%	62.49%	58.24%
	AWQ	53.22%	44.40%	67.63%	63.54%	56.87%	56.85%	45.69%	68.80%	63.66%	58.65%
	+IntactKV$_{[B]}$	53.37%	44.40%	67.50%	63.63%	**56.91%**	57.07%	45.96%	68.51%	63.63%	**58.70%**

Table 9: INT4-group128 quantization results of Vicuna models on 0-shot and 5-shot MMLU benchmarks.

the open-sourced lm-evaluation-harness[7] library for evaluation. Similar to PPL evaluation, to assess the performance of IntactKV, we prepend `[BOS]` token to the beginning of each input sequence. For the evaluation of Vicuna models, we also follow the evaluation protocol in lm-evaluation-harness and do not use a system prompt.

MT-bench. MT-bench employs a GPT-4 model to score the generated content. In our experiments, we find that the score given by GPT-4 can vary for the same content even when the generation temperature of GPT-4 is set to 0. Besides, content generation for the writing and roleplay category has a relatively high generation temperature of 0.7, which also results in variations in the final score. To faithfully assess the performance of the quantized model, we run the content generation process of each model 3 times with random seeds 42, 43, and 44. We report the mean score of three trials as the final score in Table 5 and Table 6. Also, we note that GPT-4-Turbo has been shown to be smarter than GPT-4[8], and in our experiments, we find that GPT-4-Turbo can give more stable score than GPT-4. Therefore, we evaluate the generation results on MT-bench with the latest gpt-4-0125-preview API (i.e., GPT-4-Turbo) provided by OpenAI to further reduce variations in the final score.

F More Experiment Results

F.1 PPL Results on WikiText2

As shown in Table 7, AWQ+IntactKV consistently improves AWQ for every model on the WikiText2 dataset and outperforms OmniQuant for five out of seven models. We note that OmniQuant uses the WikiText2 dataset as a calibration set, while AWQ uses Pile dataset, which may give OmniQuant a bonus for PPL evaluation on the WikiText2 dataset.

[7] https://github.com/EleutherAI/lm-evaluation-harness

[8] https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

Model	#bits	Method	OBQA	WinoGrande	ARC-C	ARC-E	BoolQ	HellaSwag	LAMBADA	Avg
Vicuna-V1.5-7B	FP16	-	45.00%	69.53%	45.73%	71.25%	80.92%	73.78%	71.12%	65.33%
	w3g128	RTN	39.40%	66.22%	43.34%	67.72%	77.40%	71.71%	61.17%	60.99%
		GPTQ	37.80%	63.30%	41.81%	64.73%	74.46%	65.97%	58.57%	58.09%
		OmniQuant	41.40%	66.22%	43.52%	67.26%	77.13%	70.35%	66.23%	61.73%
		AWQ	40.80%	67.01%	43.34%	67.55%	78.32%	71.03%	64.97%	61.86%
		+IntactKV$_{[B]}$	42.20%	67.64%	41.98%	68.52%	79.02%	71.24%	66.82%	**62.49%**
	w4g128	RTN	43.80%	68.82%	45.39%	70.33%	81.10%	73.31%	69.09%	64.55%
		GPTQ	44.40%	69.93%	44.71%	70.45%	80.76%	73.72%	70.29%	64.89%
		OmniQuant	43.00%	68.75%	44.28%	70.50%	81.01%	72.82%	70.10%	64.35%
		AWQ	43.40%	68.75%	45.39%	70.66%	81.35%	73.43%	69.47%	64.64%
		+IntactKV$_{[B]}$	44.00%	68.90%	45.90%	71.63%	82.29%	73.52%	69.61%	**65.12%**
Vicuna-v1.5-13B	FP16	-	45.40%	71.51%	50.68%	74.87%	85.29%	77.50%	73.43%	68.38%
	w3g128	RTN	42.00%	70.01%	47.44%	**72.77%**	82.20%	74.18%	69.20%	65.40%
		GPTQ	42.40%	69.53%	48.46%	71.84%	83.76%	74.48%	70.00%	65.78%
		OmniQuant	43.40%	68.51%	47.53%	71.09%	82.32%	73.92%	69.84%	65.23%
		AWQ	44.00%	68.75%	48.21%	71.76%	83.58%	75.09%	70.68%	66.01%
		+IntactKV$_{[B]}$	45.40%	70.32%	48.38%	72.14%	85.20%	75.23%	71.86%	**66.93%**
	w4g128	RTN	44.20%	70.80%	48.98%	73.82%	84.68%	76.36%	73.04%	67.41%
		GPTQ	45.80%	70.96%	50.51%	73.99%	85.47%	76.70%	73.43%	68.12%
		OmniQuant	43.80%	70.24%	49.74%	73.61%	84.59%	76.35%	72.54%	67.27%
		AWQ	44.00%	72.06%	49.15%	73.44%	85.17%	77.00%	72.77%	67.66%
		+IntactKV$_{[B]}$	45.40%	73.09%	49.57%	74.45%	85.66%	77.32%	72.75%	**68.32%**
Vicuna-V1.3-7B	FP16	-	43.80%	69.46%	44.54%	71.89%	78.07%	73.93%	69.98%	64.52%
	w3g128	RTN	41.00%	64.64%	39.08%	65.03%	75.81%	68.59%	59.73%	59.13%
		GPTQ	39.60%	64.17%	42.24%	65.95%	72.63%	70.36%	63.65%	59.80%
		OmniQuant	42.00%	67.88%	40.02%	66.75%	75.41%	70.52%	64.97%	61.08%
		AWQ	41.60%	67.96%	38.65%	65.95%	76.36%	71.27%	64.76%	60.94%
		+IntactKV$_{[B]}$	43.60%	68.43%	39.16%	67.30%	77.28%	71.20%	66.54%	**61.93%**
	w4g128	RTN	41.80%	67.80%	44.45%	69.91%	75.57%	73.25%	67.26%	62.86%
		GPTQ	44.20%	68.75%	43.60%	70.88%	75.20%	73.39%	69.42%	63.63%
		OmniQuant	42.40%	68.98%	44.03%	71.04%	76.97%	73.19%	69.49%	63.73%
		AWQ	42.80%	67.09%	43.43%	71.34%	76.36%	73.46%	68.78%	63.32%
		+IntactKV$_{[B]}$	43.80%	68.59%	42.92%	71.84%	76.79%	73.49%	69.57%	**63.86%**
Vicuna-V1.3-13B	FP16	-	45.40%	71.03%	47.70%	73.70%	82.81%	77.00%	72.91%	67.22%
	w3g128	RTN	42.20%	69.77%	44.03%	68.14%	80.52%	73.28%	64.68%	63.23%
		GPTQ	42.00%	68.35%	44.88%	70.08%	80.67%	74.24%	68.00%	64.03%
		OmniQuant	43.60%	70.56%	44.62%	70.66%	81.28%	74.13%	68.79%	64.81%
		AWQ	42.60%	68.75%	45.90%	69.57%	80.80%	74.78%	69.30%	64.53%
		+IntactKV$_{[B]}$	43.20%	69.46%	46.16%	69.74%	81.80%	75.11%	69.67%	**65.02%**
	w4g128	RTN	43.80%	71.19%	47.01%	72.64%	82.35%	76.19%	71.22%	66.34%
		GPTQ	44.60%	70.01%	47.87%	73.32%	82.23%	76.55%	71.78%	66.62%
		OmniQuant	45.80%	71.35%	46.08%	72.14%	82.35%	76.34%	71.38%	66.49%
		AWQ	44.00%	70.01%	46.67%	72.64%	82.66%	76.43%	71.74%	66.31%
		+IntactKV$_{[B]}$	45.60%	71.19%	47.10%	73.32%	82.72%	76.95%	71.38%	**66.89%**
Vicuna-V1.3-33B	FP16	-	47.80%	74.35%	51.79%	74.71%	83.91%	80.38%	73.74%	69.53%
	w3g128	RTN	46.00%	72.45%	49.83%	71.63%	82.75%	77.86%	69.82%	67.19%
		GPTQ	44.40%	72.14%	47.61%	69.65%	83.49%	77.60%	71.86%	66.68%
		OmniQuant	45.60%	73.32%	47.95%	72.52%	83.58%	78.05%	72.06%	67.58%
		AWQ	45.40%	72.77%	50.77%	72.56%	82.42%	78.33%	71.45%	67.67%
		+IntactKV$_{[B]}$	44.80%	73.56%	51.11%	72.60%	82.78%	78.55%	71.90%	**67.90%**
	w4g128	RTN	46.60%	73.95%	51.96%	74.07%	83.43%	80.01%	73.43%	69.06%
		GPTQ	47.00%	73.48%	50.85%	73.06%	83.67%	80.31%	72.50%	68.70%
		OmniQuant	47.60%	73.72%	51.96%	73.32%	83.58%	79.85%	73.57%	**69.09%**
		AWQ	46.20%	73.40%	50.60%	73.53%	84.04%	79.91%	73.39%	68.72%
		+IntactKV$_{[B]}$	45.60%	73.24%	50.94%	74.12%	84.28%	79.70%	73.14%	68.72%

Table 10: Quantization results of Vicuna models on seven 0-shot commonsense QA tasks.

F.2 MMLU Results

We provide INT3-g128 quantization results on MMLU in Table 8, and INT4-g128 quantization results on MMLU in Table 9. For INT3-g128 quantization, AWQ+IntactKV consistently improves AWQ in every experiment setting and outperforms OmniQuant for nine out of ten settings. For INT4-g128 quantization, AWQ+IntactKV leads to relatively less improvement over AWQ on the MMLU benchmark compared with INT3-g128 quantization, but still outperforms AWQ in nine out of ten experiment settings, and performs on par with OmniQuant on MMLU.

Model	Download URL
LLaMA-2-7B	https://huggingface.co/meta-llama/Llama-2-7b
LLaMA-2-13B	https://huggingface.co/meta-llama/Llama-2-13b
LLaMA-2-70B	https://huggingface.co/meta-llama/Llama-2-70b
Vicuna-v1.3-7B	https://huggingface.co/lmsys/vicuna-7b-v1.3
Vicuna-v1.3-13B	https://huggingface.co/lmsys/vicuna-13b-v1.3
Vicuna-v1.3-33B	https://huggingface.co/lmsys/vicuna-33b-v1.3
Vicuna-v1.5-7B	https://huggingface.co/lmsys/vicuna-7b-v1.5
Vicuna-v1.5-13B	https://huggingface.co/lmsys/vicuna-13b-v1.5

Table 11: Download links to officially released LLMs.

F.3 Commonsense QA Results

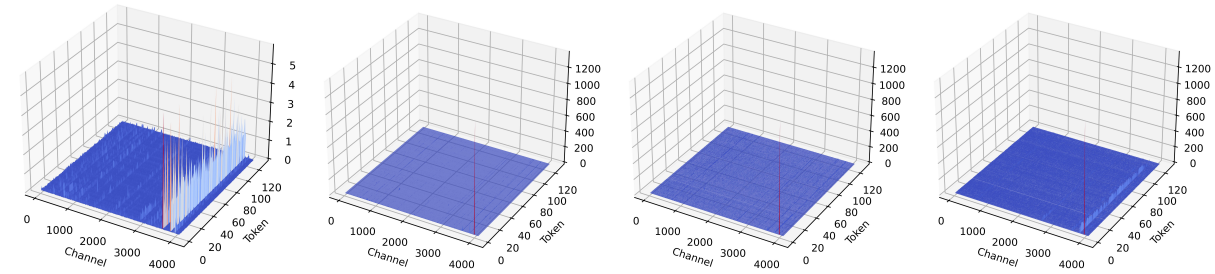
We conduct experiments on seven zero-shot commonsense QA tasks for the Vicuna family with both INT3-g128 and INT4-g128 quantization. The results are shown in Table 10. For INT3-g128 quantization, AWQ+INTACTKV significantly surpasses all baselines in all experiment settings. For INT4-g128 quantization, AWQ+INTACTKV consistently improves AWQ and outperforms OmniQuant in four out of five experiment settings, demonstrating the necessity for keeping an intact KV cache for pivot tokens.

G Experiment Details of KV Cache Quantization

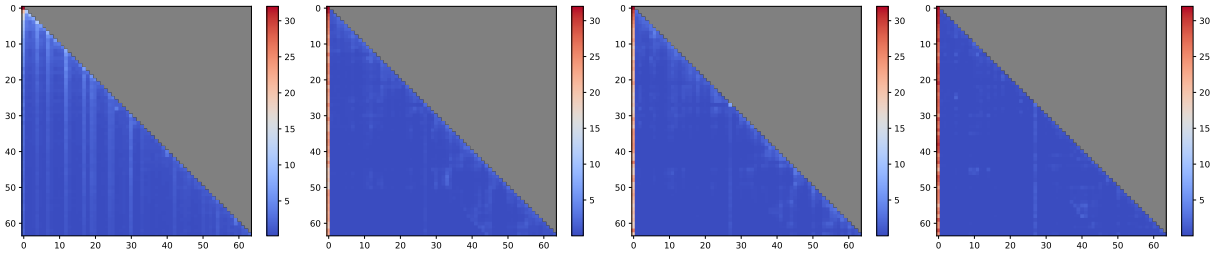
We apply asymmetric per-head dynamic quantization to the KV cache. When combined with INTACTKV, we still keep INTACTKV in FP16 while the KV cache of other tokens in low bit, which only induces negligible memory overhead compared with quantizing the entire KV cache to low bits since INTACTKV only contains the KV cache of a few tokens.

H Links to Officially Released LLMs

We provide download links to some officially released LLMs used in our experiments in Table 11.

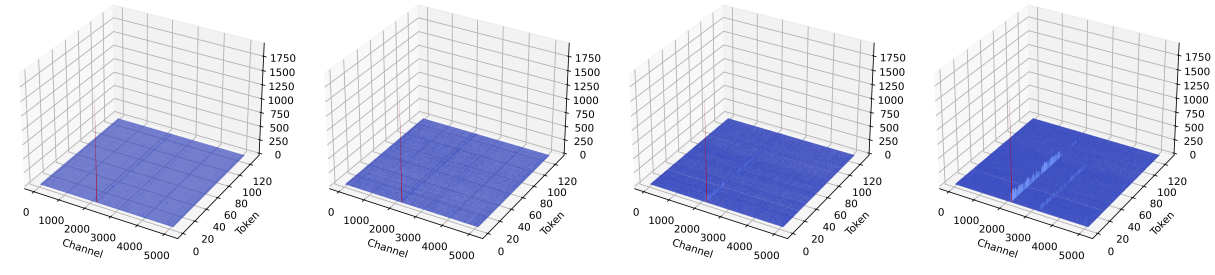


(a) Output activations of LLaMA-7B Layer 0 (b) Output activations of LLaMA-7B Layer 8 (c) Output activations of LLaMA-7B Layer 16 (d) Output activations of LLaMA-7B Layer 24

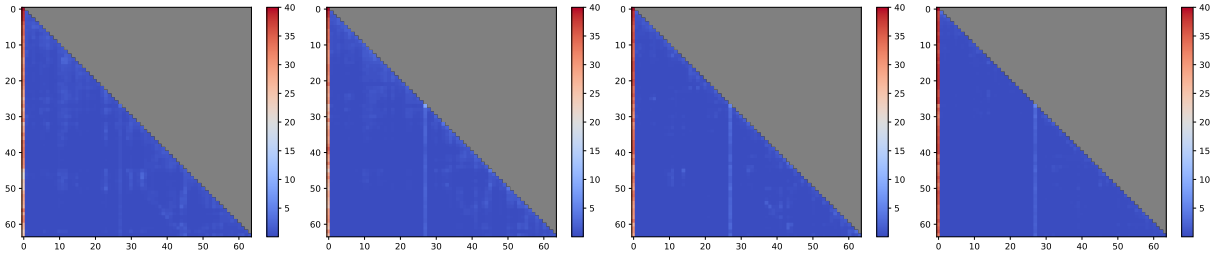


(e) Attention map of LLaMA-7B Layer 0 (f) Attention map of LLaMA-7B Layer 8 (g) Attention map of LLaMA-7B Layer 16 (h) Attention map of LLaMA-7B Layer 24

Figure 6: Magnitude of the output activations and attention map in LLaMA-7B.

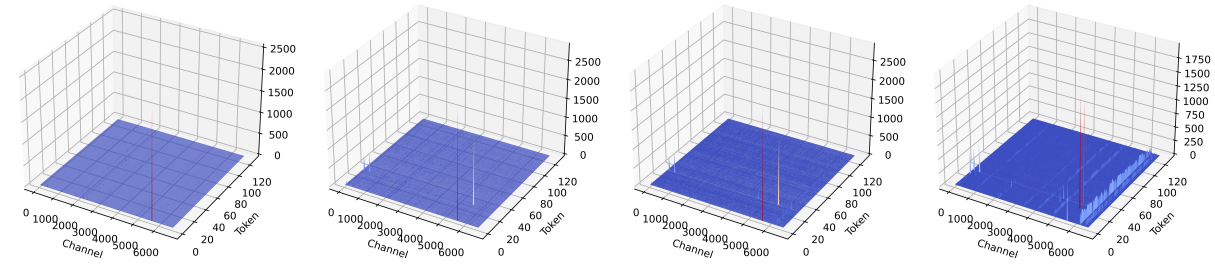


(a) Output activations of LLaMA-13B Layer 8 (b) Output activations of LLaMA-13B Layer 16 (c) Output activations of LLaMA-13B Layer 24 (d) Output activations of LLaMA-13B Layer 32

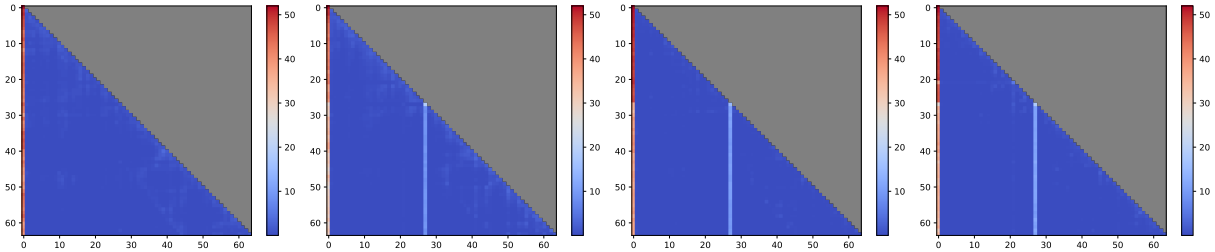


(e) Attention map of LLaMA-13B Layer 8 (f) Attention map of LLaMA-13B Layer 16 (g) Attention map of LLaMA-13B Layer 24 (h) Attention map of LLaMA-13B Layer 32

Figure 7: Magnitude of the output activations and attention map in LLaMA-13B.

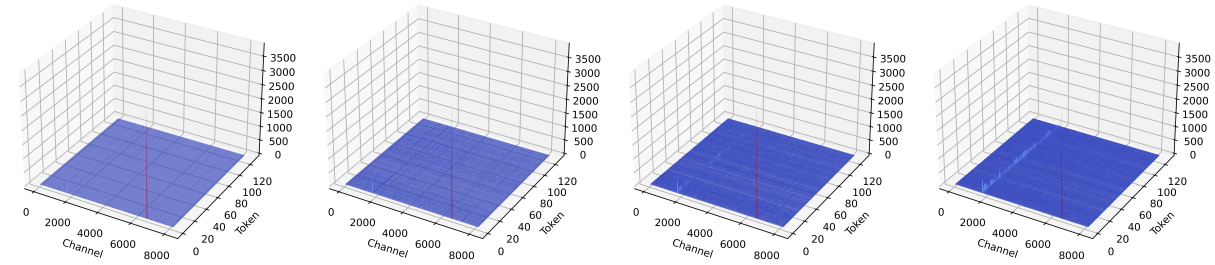


(a) Output activations of LLaMA-30B Layer 8 (b) Output activations of LLaMA-30B Layer 24 (c) Output activations of LLaMA-30B Layer 40 (d) Output activations of LLaMA-30B Layer 56

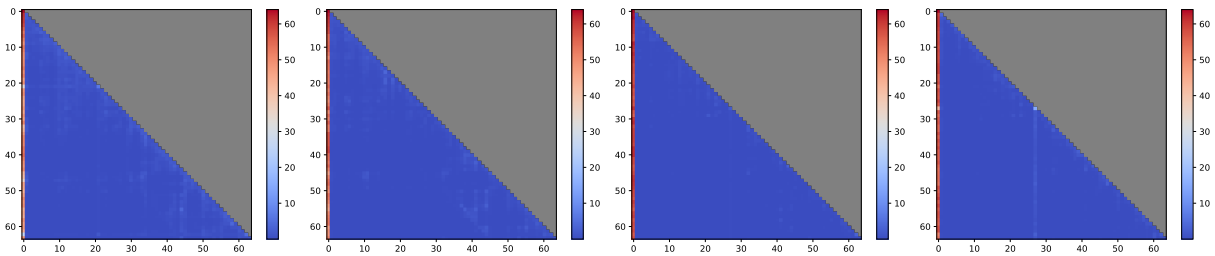


(e) Attention map of LLaMA-30B Layer 8 (f) Attention map of LLaMA-30B Layer 24 (g) Attention map of LLaMA-30B Layer 40 (h) Attention map of LLaMA-30B Layer 56

Figure 8: Magnitude of the output activations and attention map in LLaMA-30B.

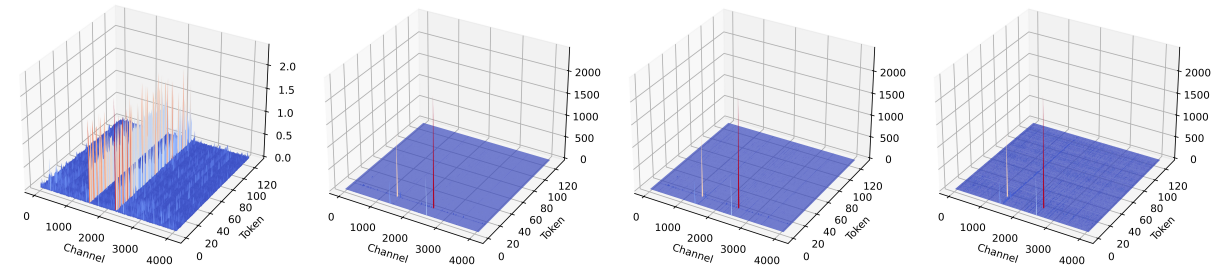


(a) Output activations of LLaMA-65B Layer 16 (b) Output activations of LLaMA-65B Layer 32 (c) Output activations of LLaMA-65B Layer 48 (d) Output activations of LLaMA-65B Layer 64

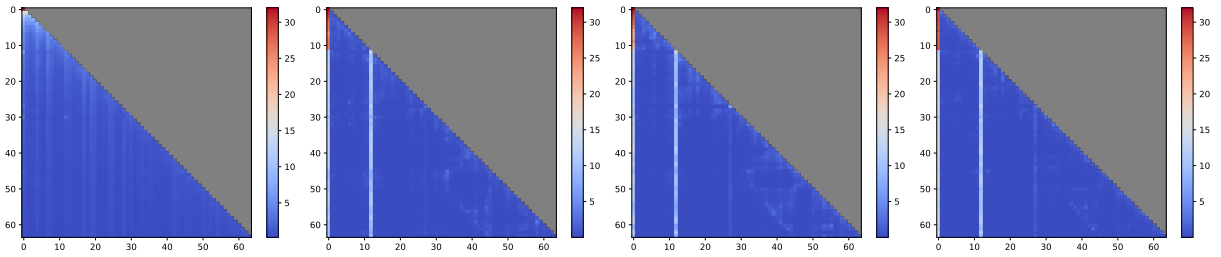


(e) Attention map of LLaMA-65B Layer 16 (f) Attention map of LLaMA-65B Layer 32 (g) Attention map of LLaMA-65B Layer 48 (h) Attention map of LLaMA-65B Layer 64

Figure 9: Magnitude of the output activations and attention map in LLaMA-65B.

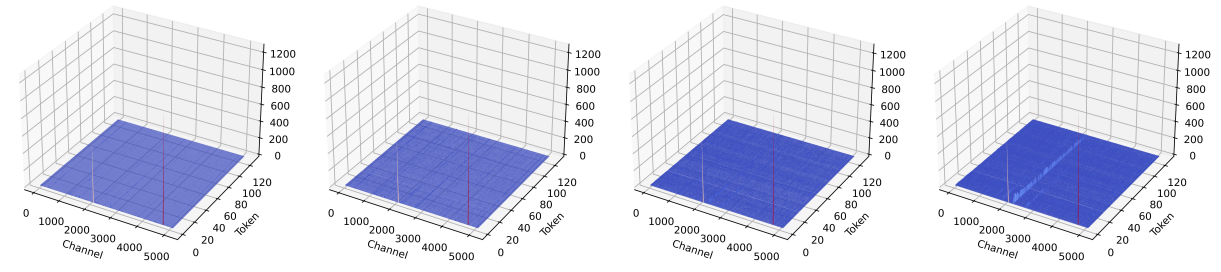


(a) Output activations of LLaMA-2-7B Layer 0 (b) Output activations of LLaMA-2-7B Layer 8 (c) Output activations of LLaMA-2-7B Layer 16 (d) Output activations of LLaMA-2-7B Layer 24

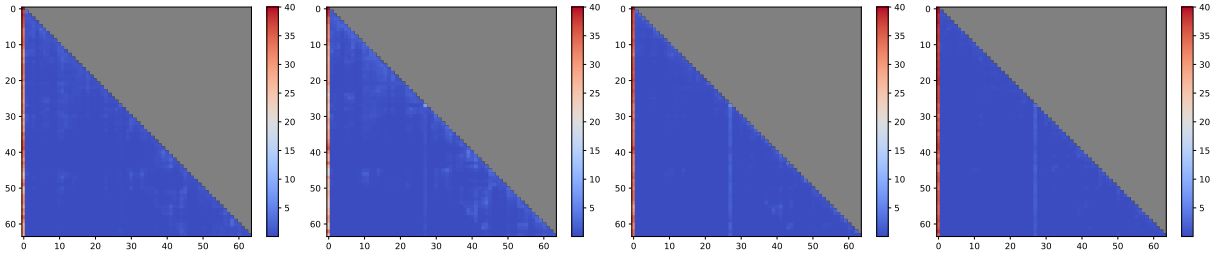


(e) Attention map of LLaMA-2-7B Layer 0 (f) Attention map of LLaMA-2-7B Layer 8 (g) Attention map of LLaMA-2-7B Layer 16 (h) Attention map of LLaMA-2-7B Layer 24

Figure 10: Magnitude of the output activations and attention map in LLaMA-2-7B.

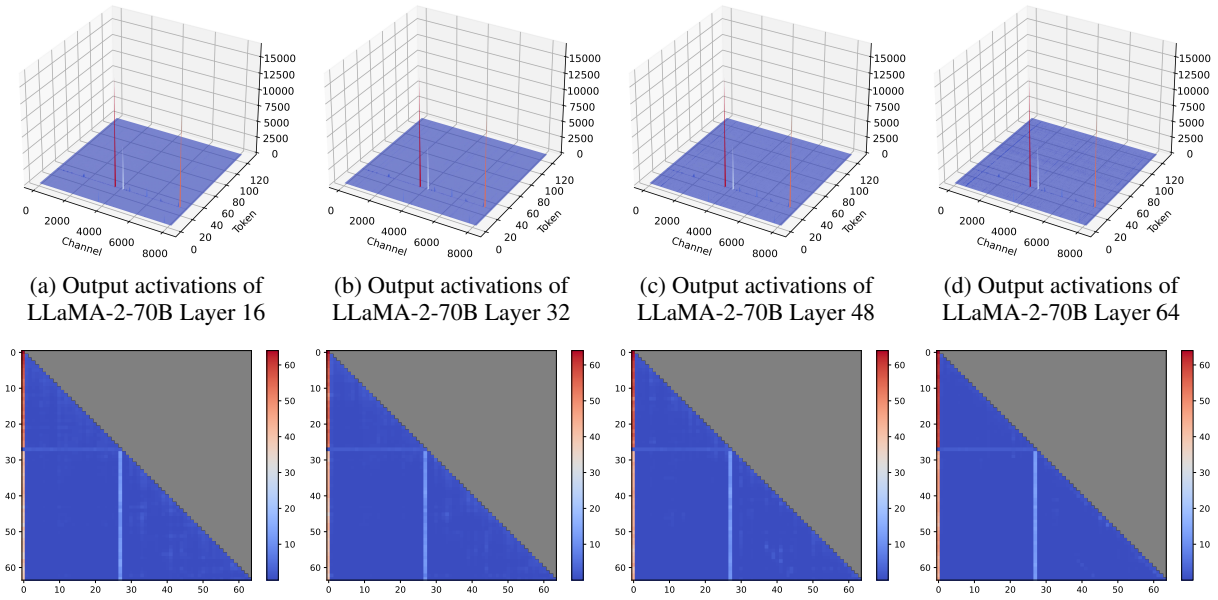


(a) Output activations of LLaMA-2-13B Layer 8 (b) Output activations of LLaMA-2-13B Layer 16 (c) Output activations of LLaMA-2-13B Layer 24 (d) Output activations of LLaMA-2-13B Layer 32



(e) Attention map of LLaMA-2-13B Layer 8 (f) Attention map of LLaMA-2-13B Layer 16 (g) Attention map of LLaMA-2-13B Layer 24 (h) Attention map of LLaMA-2-13B Layer 32

Figure 11: Magnitude of the output activations and attention map in LLaMA-2-13B.



(a) Output activations of LLaMA-2-70B Layer 16

(b) Output activations of LLaMA-2-70B Layer 32

(c) Output activations of LLaMA-2-70B Layer 48

(d) Output activations of LLaMA-2-70B Layer 64

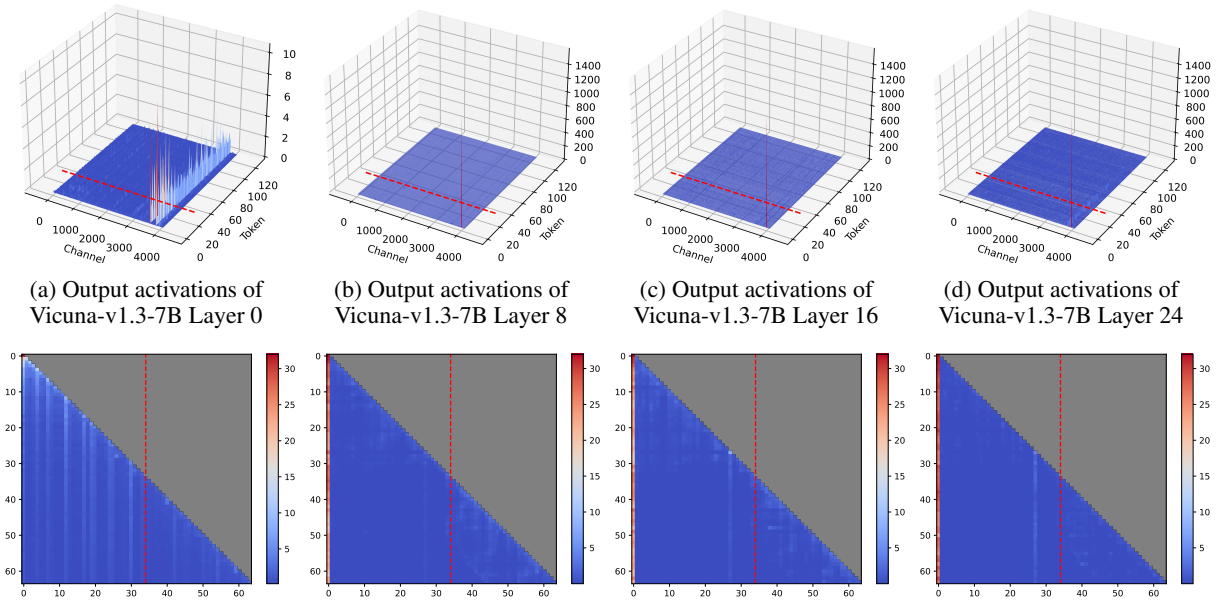
(e) Attention map of LLaMA-2-70B Layer 16

(f) Attention map of LLaMA-2-70B Layer 32

(g) Attention map of LLaMA-2-70B Layer 48

(h) Attention map of LLaMA-2-70B Layer 64

Figure 12: Magnitude of the output activations and attention map in LLaMA-2-70B.



(a) Output activations of Vicuna-v1.3-7B Layer 0

(b) Output activations of Vicuna-v1.3-7B Layer 8

(c) Output activations of Vicuna-v1.3-7B Layer 16

(d) Output activations of Vicuna-v1.3-7B Layer 24

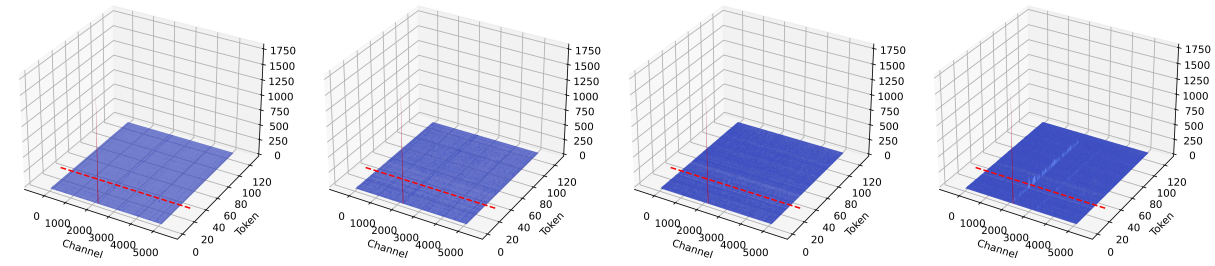
(e) Attention map of Vicuna-v1.3-7B Layer 0

(f) Attention map of Vicuna-v1.3-7B Layer 8

(g) Attention map of Vicuna-v1.3-7B Layer 16

(h) Attention map of Vicuna-v1.3-7B Layer 24

Figure 13: Magnitude of the output activations and attention map in Vicuna-v1.3-7B. The tokens before the red dashed line correspond to the Vicuna system prompt.

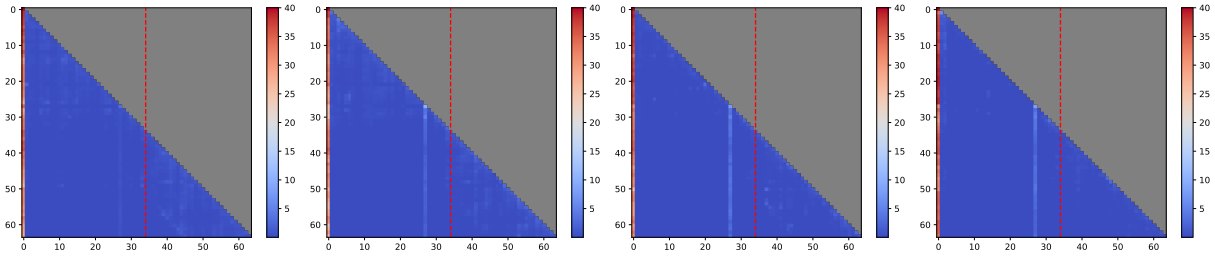


(a) Output activations of Vicuna-v1.3-13B Layer 8

(b) Output activations of Vicuna-v1.3-13B Layer 16

(c) Output activations of Vicuna-v1.3-13B Layer 24

(d) Output activations of Vicuna-v1.3-13B Layer 32



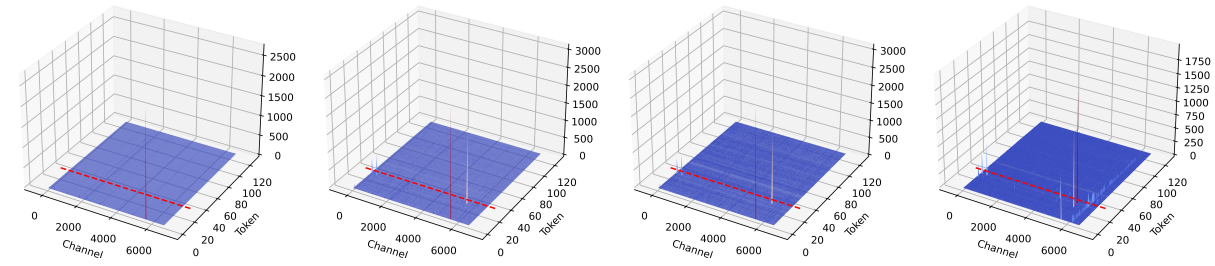
(e) Attention map of Vicuna-v1.3-13B Layer 8

(f) Attention map of Vicuna-v1.3-13B Layer 16

(g) Attention map of Vicuna-v1.3-13B Layer 24

(h) Attention map of Vicuna-v1.3-13B Layer 32

Figure 14: Magnitude of the output activations and attention map in Vicuna-v1.3-13B. The tokens before the red dashed line correspond to the Vicuna system prompt.

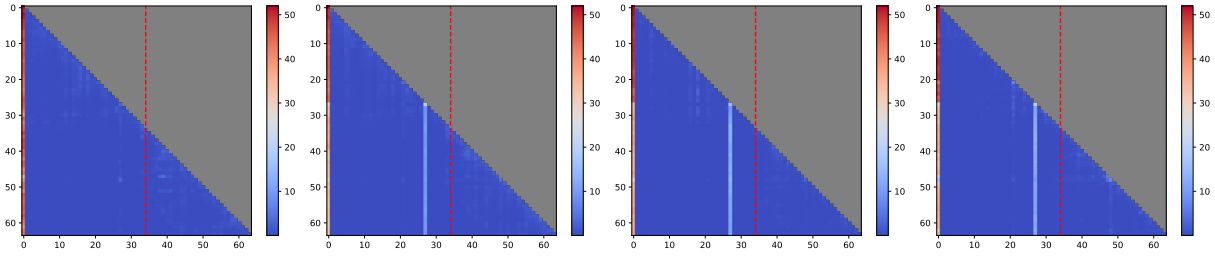


(a) Output activations of Vicuna-v1.3-33B Layer 8

(b) Output activations of Vicuna-v1.3-33B Layer 24

(c) Output activations of Vicuna-v1.3-33B Layer 40

(d) Output activations of Vicuna-v1.3-33B Layer 56



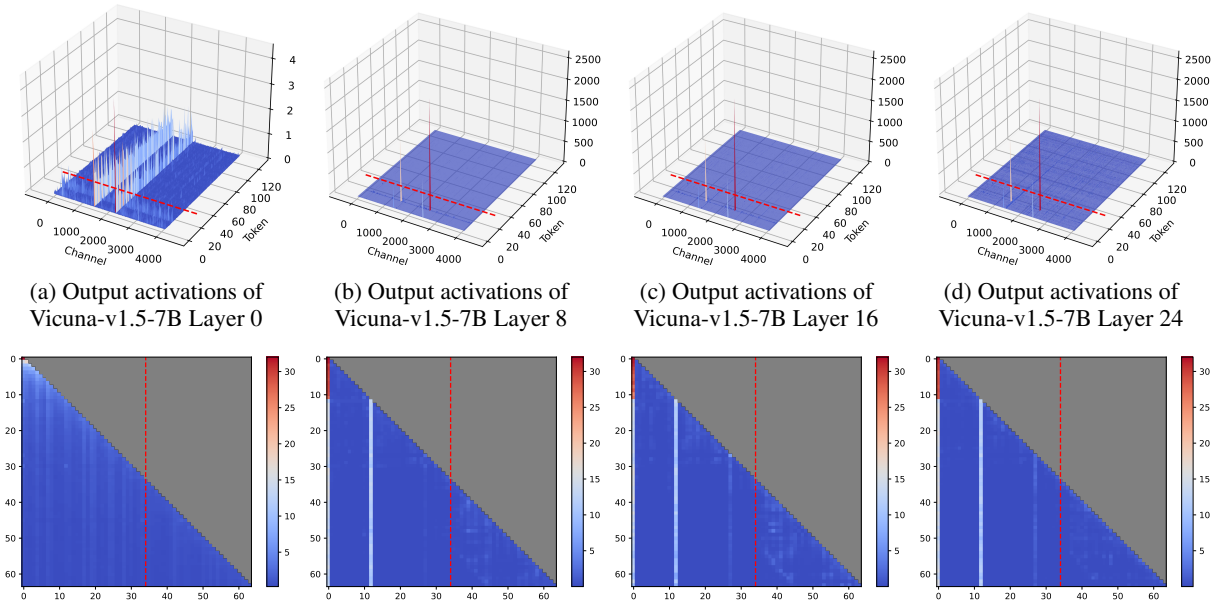
(e) Attention map of Vicuna-v1.3-33B Layer 8

(f) Attention map of Vicuna-v1.3-33B Layer 24

(g) Attention map of Vicuna-v1.3-33B Layer 40

(h) Attention map of Vicuna-v1.3-33B Layer 56

Figure 15: Magnitude of the output activations and attention map in Vicuna-v1.3-33B. The tokens before the red dashed line correspond to the Vicuna system prompt.



(a) Output activations of Vicuna-v1.5-7B Layer 0

(b) Output activations of Vicuna-v1.5-7B Layer 8

(c) Output activations of Vicuna-v1.5-7B Layer 16

(d) Output activations of Vicuna-v1.5-7B Layer 24

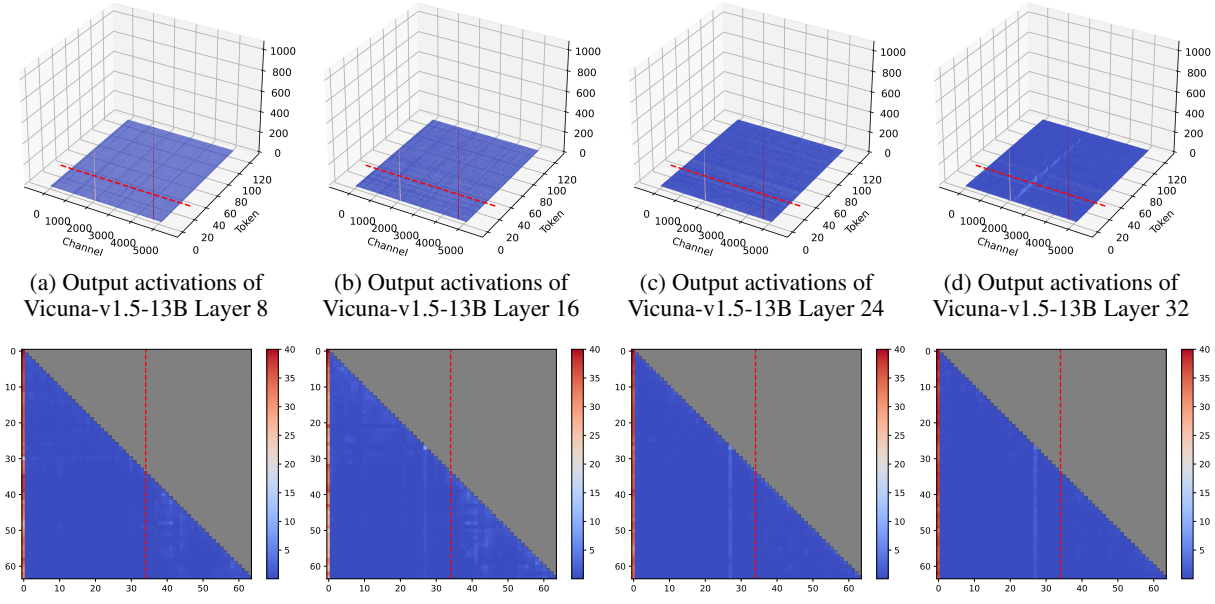
(e) Attention map of Vicuna-v1.5-7B Layer 0

(f) Attention map of Vicuna-v1.5-7B Layer 8

(g) Attention map of Vicuna-v1.5-7B Layer 16

(h) Attention map of Vicuna-v1.5-7B Layer 24

Figure 16: Magnitude of the output activations and attention map in Vicuna-v1.5-7B. The tokens before the red dashed line correspond to the Vicuna system prompt.



(a) Output activations of Vicuna-v1.5-13B Layer 8

(b) Output activations of Vicuna-v1.5-13B Layer 16

(c) Output activations of Vicuna-v1.5-13B Layer 24

(d) Output activations of Vicuna-v1.5-13B Layer 32

(e) Attention map of Vicuna-v1.5-13B Layer 8

(f) Attention map of Vicuna-v1.5-13B Layer 16

(g) Attention map of Vicuna-v1.5-13B Layer 24

(h) Attention map of Vicuna-v1.5-13B Layer 32

Figure 17: Magnitude of the output activations and attention map in Vicuna-v1.5-13B. The tokens before the red dashed line correspond to the Vicuna system prompt.