

Mitigate Position Bias in LLMs via Scaling a Single Hidden States Channel

Yijiong Yu^{1†}, Huijiang Jiang², Xufang Luo², Qianhui Wu², Chin-Yew Lin², Dongsheng Li², Yuqing Yang², Yongfeng Huang¹, Lili Qiu²

¹Tsinghua University, ²Microsoft Corporation

yuyj22@mails.tsinghua.edu.cn, yfhuang@tsinghua.edu.cn

{hjiang, xufluo, qianhuiwu, cyl, dongsli, yuqyang, liliqiu}@microsoft.com

Abstract

Long-context language models (LCLMs) can process long context, but still exhibit position bias, also known as “lost in the middle”, which indicates placing key information in the middle of the context will significantly affect performance. To mitigate this, we first explore the micro-level manifestations of position bias, concluding that attention weights are a micro-level expression of position bias. Then we identify that, in addition to position embeddings, positional information in hidden states also contributes to position bias, and it manifests itself in specific channels of hidden states, called positional hidden states. Based on these, we propose a method to mitigate position bias by scaling positional hidden states. Experiments on NaturalQuestions Multi-document QA, KV retrieval and LongBench, using various models including RoPE models, context window-extended models, and Alibi models, demonstrate the effectiveness and generalizability of our approach. Our method can improve performance by up to 15.2% in “lost in the middle” benchmark by modifying just one channel of hidden states. Our code is available at <https://aka.ms/PositionalHidden>.

1 Introduction

Long-context language models (LCLMs) (Reid et al., 2024; Liu et al., 2025; Young et al., 2024; Abdin et al., 2024; DeepSeek-AI, 2024) have recently garnered significant attention within the community, enabling LLMs to handle longer and more complex tasks such as long-context question-answering (Caciularu et al., 2023; Li et al., 2025). However, recent research (Li et al., 2025; Liu et al., 2024; Li et al., 2024; Shi et al., 2023; Tang et al., 2023; He et al., 2024a; Zhang et al., 2024) shows that even long-context LLMs often fail to utilize all context information effectively, exhibiting a “lost in the middle” bias where middle context information

is ignored. This issue affects all types of LLMs, regardless of their architecture or size, and worsens with longer contexts.

Previous works have analyzed this issue from the perspectives of data distribution (He et al., 2024a; Yu, 2023; An et al., 2024) and position embeddings (Zhang et al., 2024; Chen et al., 2024). For example, FILM (An et al., 2024) addresses position bias by constructing data with key information distributed in various positions for supervised fine-tuning (SFT). Ms-PoE (Zhang et al., 2024) mitigates position bias by interpolating RoPE (Su et al., 2024) using head-wise scaling factors.

On the other hand, some works (Haviv et al., 2022; Wang et al., 2024; Chi et al., 2023) have proven that, besides position embeddings, the hidden states of LLMs can also convey positional information, which is generated by the causal attention mask. Although various works (see related works in Appendix A) have attempted to mitigate position bias, few of them have related position bias to positional information in hidden states, which may be another non-negligible source of position bias.

To verify the relation between position bias and positional information in hidden states, we conduct a series of experiments. First, as many previous works (Yin et al., 2024; Wang et al., 2025; Chen et al., 2024) have observed, we find U-shaped attention weights consistent with position bias in specific layers. Second, we use perturbation experiments to prove position bias is indeed affected by positional information in hidden states. Third, We identify some channels called “positional channel” (the hidden states corresponding to them are called “positional hidden states”), whose values can manifest absolute positional information, through a process of hypothesis and subsequent verification. Finally, we confirm these channels can indeed affect position bias.

Naturally, to mitigate position bias, a direct way is to eliminate (or minimize) its potential cause:

[†]Work during internship at Microsoft.

positional hidden states. So, we propose a position bias mitigation method named “**scale positional hidden states**”. Specifically, we first design a heuristic searching algorithm that quickly identifies which channels (along the hidden size axis) of the hidden states are potential positional channel, using monotonicity and smoothness as indicators. Then We select the optimal channel among them based on the loss on a calibration dataset. Next, based on extensive engineering experience, we design a well-designed attention modification algorithm that let the scaled hidden states only influence the attention queried by the last token, to eliminate positional bias in attention, while avoiding interfering too much with the original attention leads to unstable model performance.

Extensive experiments on various models, including LLaMA-2 (Touvron et al., 2023), Vicuna (Chiang et al., 2023), Mistral (Jiang et al., 2023), Gemma (Team et al., 2024), Qwen (Bai et al., 2023), and MPT (Team, 2023), and across different tasks, including Multi-document QA, KV retrieval, LongBench (Bai et al., 2024) benchmark, demonstrate that our method effectively mitigates position bias by modifying only one channel of the hidden states of the model. In addition, we test on timeline reorder task (Li et al., 2024) and MMLU (Hendrycks et al., 2021) to show our methods has minimal side effects on the original capabilities of the model.

Our main contributions are as follows:

1. We are the first to discover that the positional information in hidden states is also the cause of position bias, especially the bias to the beginning.
2. We are the first to find explicit, visible hidden states channels which are approximately correlated to absolute token positions, called positional hidden states. And we confirm they can affect position bias.
3. We propose a method for identifying and scaling the positional hidden states to mitigate position bias.

2 Positional Information in hidden states affects position bias

In this section, we first identify patterns in attention weights that closely correspond to position bias. Then, through modifying the causal mask in former layers and observing the change of attention

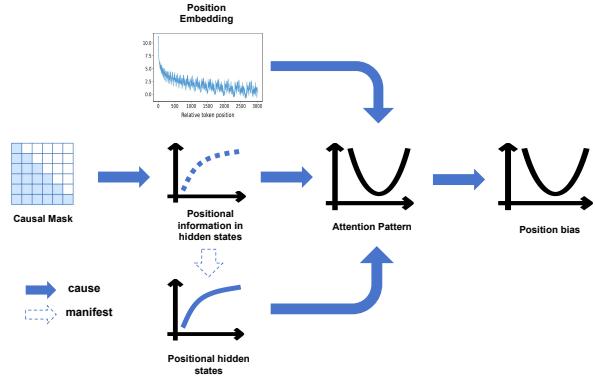


Figure 1: The relationship between causal mask, positional information in hidden states, positional hidden states, position embedding, attention pattern and position bias of model’s performance.

in latter layers, we demonstrate position-related attention bias can also be affected by the positional information in hidden states. Third, we extract specific hidden states channels which has monotonicity with the token position, named “positional hidden states”, and find that they bears responsibility for the emergence of position bias. To sum up, we show our findings about position bias in this section in Figure 1.

2.1 Microscopic Manifestations of Position Bias in Transformers: Attention Weight Patterns

To explore the micro-level manifestations of position bias in Transformers, we analyze the attention weights in a classic long-context task, KV retrieval, which requires the model to retrieve the value of the given key from a list containing 50 Key-Value pairs (see Appendix D.1 for detailed prompts). The KV pair corresponding to the given key in the question is called the **Gold KV**. In attention analysis, we average all the attention weights from the last token of the question to the tokens of the i -th KV pair as the model’s “attention to the i -th KV”. More details about how we calculate attention are in Appendix D.1.

As shown in Figures 2, in deep layers, the model exhibits retrieval-like behavior, focusing on key information, forming a diagonal pattern observed in Figure 2b. So we call them **retrieval-related layers**. While in other shallow layers, it always focus most attention on the start or end of the prompt, wherever the key information is located, exhibiting vertical lines patterns, as shown in Figure 2a. Attention patterns of all the layers are shown in Appendix F, based on which we roughly regard that

layers 15~20 belong to the retrieval-related layers.

In retrieval-related layers, it can be observed that the attention weights for key information (Gold KV) exhibit patterns similar to position bias: when key information is located at the start or end of the prompt, the attention weights focused on it are relatively higher, while in the middle, lower. Moreover, we extract the attention to key information (average of layers 15~20) with different context length. As shown in Figure 2c, as the context length grows, the attenuation of attention weights with respect to position becomes more pronounced.

Furthermore, in Appendix D.2, we find artificially adjusting the attention weights to the key information can directly improve the corresponding retrieval accuracy (this is intuitive). Thus, we confirm that position bias of model performance is to a large extent caused by the bias in attention weight patterns. So in the following experiments, we also use attention weights as the indicator of whether the model has successfully retrieved the key information, as well as the retrieval accuracy.

2.2 Positional Information in Hidden States Also Contributes to Position Bias

Recent works (Haviv et al., 2022; Wang et al., 2024; Chi et al., 2023) have indicated that, besides position embeddings, the causal mask can also introduce positional information, which is then stored in the hidden states of LLMs. However, there is no research about whether such information will affect position bias. Therefore, we design a perturbation experiment using Mistral-7B-v0.2 (32 layers) on the classic KV retrieval task, where we modify the causal mask, aiming at changing position information in hidden states, and then observe whether position bias is affected.

In the experiment, only the causal mask of the layers 2 to 8 are modified, but we mainly observe the attention change in retrieval-related layers, specifically, the average attention of layers 15 to 20 is calculated. This setting avoids modifying the causal mask directly shifts the attention, letting us only see the effect of the positional information in hidden states.

The modification to the causal mask is called **Crop Mask**, which alters it so the tokens of the gold KV pair can only see itself, but not previous tokens (details about this operation are in Appendix D.3). Based on the theory of previous works, we posit this modification will make the positional feature of the tokens of the gold KV pair similar to

that of the first KV pair.

In addition, as a comparison, we also shift the position embedding of the tokens of the gold KV (in every layer) as comparisons. Such shift includes: (1) **PE to Beginning**, which assigns the position IDs of the first KV pair to that of the gold KV pair; (2) **PE to End**, which assigns the position IDs of the last (the 50th) KV pair to it.

As shown in Figure 3, the original model apparently exhibits a “lost in the middle” pattern: when gold KV is in the middle position, the KV retrieval accuracy as well as the attention to it is much lower.

The most notable result is, modifying the causal mask effectively enhances the attention to the gold KV, as well as the retrieval accuracy, whatever its position is. It even lets the attention at the middle be improved to almost on par with the beginning.

As for PE modification, “PE to end” has a certain degree of improving attention to the gold KV, but can hardly allow the model’s performance to match the accuracy when the gold KV pair is positioned at the start or end of the prompt. In contrast, “PE to Beginning” results in a noticeable performance drop as well as attention weight reduction when the gold KV is originally close to the end.

This phenomenon indicates us that, the direct cause of the attention improvement in retrieval-related layers, can only be the information transmitted through hidden states (because PE and the causal mask in these layers are both not modified). That is, besides position embedding, the positional information in hidden states, which is introduced by causal mask, is also an important factor affecting position bias.

2.3 Position Information can be clearly manifested in Specific Hidden states Channels

Definition 2.1 (Positional Hidden States). *Let $h_k(p)$ denote the k -th dimension of the hidden states across each token’s position p . We define positional hidden states h_t as hidden states whose values change consistently and monotonically with the token position. Therefore, their derivative (after curve fitting) should always be positive or negative:*

- $h'_t(p) > 0, \forall p$ or $h'_t(p) < 0, \forall p$

We have found that the positional information in hidden states can affect position bias, and previous works (Haviv et al., 2022) have proven positional information can be linearly probed from

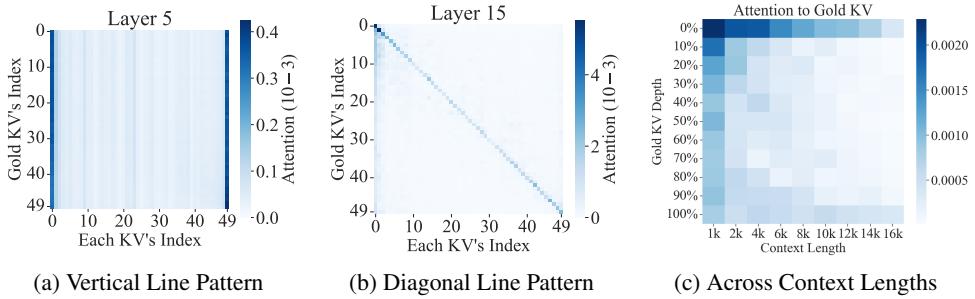


Figure 2: Attention distribution of the gold KV pair to each KV pair across different positions on the KV retrieval task (Liu et al., 2024) using Mistral-7B (Jiang et al., 2023). (a) and (b) show the results averaged across all heads of the layer. (c) shows the attention of the ground-truth KV to the ground-truth KV (i.e., diagonal lines from (b)) across different context lengths.

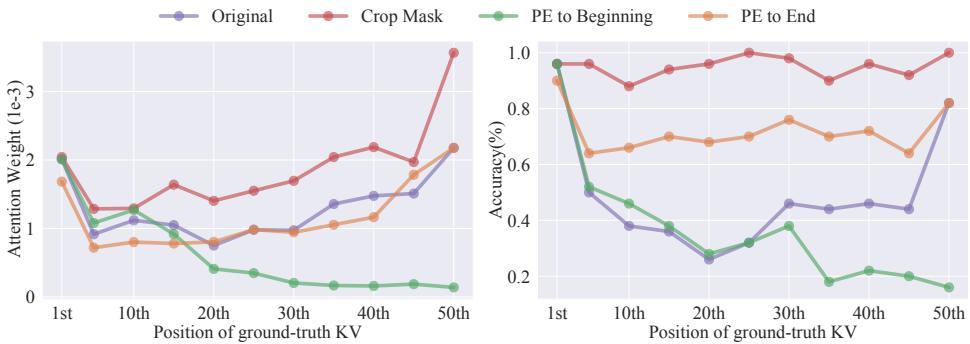


Figure 3: Performance and attention of different methods with the ground-truth KV at different positions in the KV retrieval task (Liu et al., 2024) using Mistral-7B (Jiang et al., 2023).

hidden states. However, the high-dimensional hidden states make it hard to concentrate on certain low-dimensional subspace to directly perceive or manipulate positional information in LLMs. To make this easier, we make a strong assumption: the positional information (at least part of it) in the hidden states should be manifested as activation values linearly correlated with each token’s absolute position. Thus, on average, there should exist some channels whose values change roughly monotonically with the absolute token positions, as shown in Definition 2.1. We call those channels **positional channels** and the parts of hidden states belonging to these channels as **positional hidden states**.

Based on this assumption, to identify positional channels, we average the model’s hidden states using 2000 randomly generated strings as inputs, whose lengths are 1000, to make an average hidden states with shape [*sequence length, hidden size*]. Then we traverse each channel (along *hidden size* axis, in other words, *hidden size* is the number of channels) to check whether its activation values (a one-dimensional array of *sequence length*) are

changing monotonically with token positions. To eliminate the effect of outliers and noise, we apply sliding window average with window size of 100 to that array and discard the first 30 tokens (because the hidden states values of these tokens are often too huge (Sun et al., 2024)) before checking monotonicity.

As the results shown in Figure 4, even though our assumption is very strong, our experiments reveal that causal LMs consistently possess such hidden states across most layers (details in Appendix G), regardless of whether the model has position embeddings. We further demonstrate that position hidden states are mainly determined by the causal mask but not position embeddings through perturbation experiments in Appendix D.4.

Finally, we aim to confirm that even a single position channel can significantly affect position bias. So we conduct an experiment on KV retrieval, where we manually modify (subtract a fixed value from them) the values in the 213rd hidden states channel of Mistral-7b of a KV pair, and find the attention to this KV indeed greatly increases, which proves our posit. Details about this experiment are

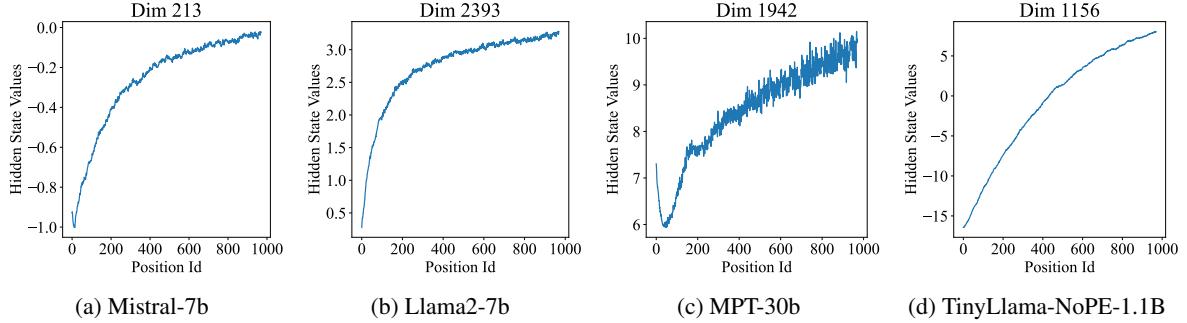


Figure 4: Hidden states values with the token positions of the positional channel averaged across all layers.

in Appendix D.5.

3 Mitigating Position Bias

Previous work focusing on PE often attempted to refine the application of RoPE, essentially minimizing the gap in position information (introduced by RoPE) between tokens in different positions.

In Section 2, we have identified positional information in hidden states as another factor causing position bias, and found positional channels strongly related to positional information. Thus, similarly, to minimize the gap in position information (introduced by positional information in hidden states) between tokens, the most direct way is to scale the activation values of those channels.

Therefore, we propose a method to mitigate position bias by scaling the positional hidden states, as shown in Figure 5. Specifically, it consists of two steps: identifying the positional hidden states h_t and scaling (multiplying) them by a factor s .

3.1 Problem Formulation

Given a pre-trained LLM θ and a general dataset $\{\mathbf{x}, \mathbf{y}\}$, our objective is to find the optimal positional hidden states h_t , i.e. the t -th channel of the hidden states, and the corresponding scaling factor s to maximally reduce position bias, which can be formulated as follows:

$$\arg \min_{h_t \in \mathcal{H}, s < 1} \frac{1}{|\mathbf{P}|} \sum_{i=1}^{|\mathbf{P}|} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{p}_i; F(\theta, h_t, s)) \quad (1)$$

where \mathbf{P} represents the set of different positions p_i of the key information within the context of the prompt \mathbf{x} , $F(\theta, h_t, s)$ denotes the operation of scaling the LLM θ on the t -th channel ($t \in [1, \text{hidden size}]$) of its hidden states by the scaling factor s (the specific scaling method is in Section 3.3), and \mathcal{L} denotes the loss for general downstream tasks of the modified model.

In practice, due to the computing cost, the loss is computed on a small validation dataset of some representative tasks. Intuitively, we can traverse all different values of t and obtain the loss on a calibration (validation) dataset to find the optimal one. However, due to the large *hidden size* which is typically over 4k, traversal search is too time-consuming, taking nearly 3 days for a 7B LLM. So, we design a search algorithm, mainly based on the monotonicity feature mentioned in Section 2, to first select a small set (no more than 50) of channels that are most likely to carry positional information that can affect position bias. Then we only need to traverse the small set of candidates to obtain their losses.

3.2 Identifying Positional Hidden States

We have defined positional hidden states in Definition 2.1, but it is just an ideal situation. In practice, because the hidden states is mainly determined by specific input text, the original values of them will not strictly satisfy monotonicity, which means we cannot rely on strict monotonicity to select the positional channels, but whether it roughly conforms to monotonicity. Thus, we use least square polynomial fit to approximate the values in the channel. Moreover, hidden states of different layers also vary. So we consider this channel as roughly monotonic if the fitted curve conforms to monotonicity in more than a quarter of all the layers of the model.

Using curve fitting, we can usually identify dozens or hundreds of channels that exhibit various degrees of monotonicity. However, some of them are very close to the ideal monotonic curve, while others fluctuate violently although they are roughly monotonic. So we use another indicator, smoothness, to evaluate if how close they are to the ideal monotonic change. The smoothness score is calculated by $\int |h''_t(p)|^2$, where $h''_t(p)$ is the sec-

Algorithm 1 Positional Hidden State Search

```

1: Input: LLM  $\theta$ , hidden states  $\mathcal{H}$ , layer number
    $L$ , validation set  $\mathcal{D}_{\text{val}}$ , positions set  $\mathcal{P}$ , thresh-
   old  $\varepsilon$ 
2: # Identify top-K positional dimensions
3:  $\rho \leftarrow \phi$ 
4: for  $t \leftarrow 1$  to  $|\mathcal{H}|$  do
5:    $c_t \leftarrow 0$ ,  $g_t \leftarrow 0$ 
6:   for  $l \leftarrow 1$  to  $L$  do
7:     if  $h'_t(p) > 0$ ,  $\forall p$  or  $h'_t(p) < 0$ ,  $\forall p$  then
8:        $c_t \leftarrow c_t + 1$ ,  $g_t \leftarrow g_t + \text{Smooth}(h'_t)$ 
9:     end if
10:    end for
11:    if  $c_t > \varepsilon$  then
12:       $\rho \leftarrow \rho \cup \{t\}$ 
13:    end if
14:  end for
15:   $\rho \leftarrow \arg \min_{t \in \rho} g_t$ 
16: # Evaluate on the validation dataset
17: for  $t \in \rho$  do
18:    $\mathcal{L}_t \leftarrow 0$ 
19:   for  $p \in \mathcal{P}$  do
20:      $\mathcal{L}_t \leftarrow \mathcal{L}_t + \mathcal{L}(\mathbf{x}, \mathbf{y}, p; F(\theta, h_t, s))$ 
21:   end for
22:    $\bar{t} \leftarrow \arg \min_{t \in \rho} \mathcal{L}_t$ 
23: return  $\bar{t}$ 

```

ond derivative (or difference) of $h(p)$, and a smaller score means smoother. Only when h_t is considered roughly monotonic, its smoothness score will be calculated, and averaged across layers. Then we only maintain the channels with the Top- K smoothness score. K is set to 10 in our default setting.

Finally, we evaluate the average loss across K channels using a 100-sample calibration dataset for KV retrieval. The channel with the lowest loss is chosen for scaling.

To determine the optimal scale factor, we perform a grid search over $\{0.5, 0, -0.5, -1\}$, selecting the factor with the lowest loss.

We organize our search algorithm in Algorithm 1, the search process consists of the following two steps: 1) Identify all the channels ρ of the hidden states that are roughly monotonic in more than ε layers, and select the top- K channels with the lowest smoothness score. Here c_t is the number of layers where $h_t(p)$ is monotonic, and g_t is the smooth score of $h_t(p)$. 2) Use a small calibration dataset $\mathcal{D}_{\text{val}} = \{\mathbf{x}, \mathbf{y}\}$ to evaluate the impact of scaling these positional hidden states respectively and select the channel $h_{\bar{t}}$ that can lead to the minimal loss $\mathcal{L}_{\bar{t}}$.

3.3 Scaling Positional Hidden States

In our early experiments we find simply scaling the positional channel of the hidden states in every layer or in every module of the model can lead to unstable performance of the model. Therefore, to minimize the side-effect of scaling positional hidden states, our proposed method is finely designed to scale the positional hidden states only affecting the attention weights from last token of the sequence, as shown in Figure 5.

Due to the inherent lack of interpretability in the working mechanisms of LLMs, our design is mainly based on the experience of trial and error experiments. So, we use some ablation experiments, shown in Appendix E, to explain why we scale only 1 channel but not more, modify the last token’s attention but not all tokens, and use 3 indicators to search the optimal channel.

Specifically, in our method, for the tokens preceding the last token, the attention calculation remains the same as the original. In a sequence of length l , for the last token’s attention computation, we obtain the modified query state $\bar{\mathbf{q}}_l$ (of the l -th token, i.e. the last token) and key states $\bar{\mathbf{K}}$ (of all the tokens) by scaling the positional hidden states. That is,

$$\begin{aligned} \bar{\mathbf{q}}_l &= \mathcal{P}(W^Q f(\mathbf{h}(l), p, s), l) \\ \bar{\mathbf{K}} &= \mathcal{P}(W^K f(\mathbf{h}, p, s), [1, 2, \dots, l]) \end{aligned} \quad (2)$$

Here $f(\mathbf{h}, p, s)$ means the p -th channel of \mathbf{h} is scaled by the factor s . Therefore, the combined attention calculation is as follows:

$$z = \begin{cases} \text{Softmax}\left(\frac{\mathbf{q}_i \mathbf{K}^\top + \text{Mask}}{\sqrt{d}}\right) \mathbf{V}, & i < l \\ \text{Softmax}\left(\frac{\bar{\mathbf{q}}_l \bar{\mathbf{K}}^\top}{\sqrt{d}}\right) \mathbf{V}, & i = l \end{cases} \quad (3)$$

where z is the attention output.

Except calculating the combined attention weights, the other modules remain the same as those in the original method. We implement our method using FlashAttention-2 (Dao, 2024) with minimal overhead, so our approach results in only a slight increase in latency, as shown in Appendix C.4.

Moreover, we find modifying the hidden states in the initial layers (especially the 1st or 2nd layer) will also make the model’s output more unstable, as well as the last layers. Thus we only apply our scaling to layers in the middle (they are more

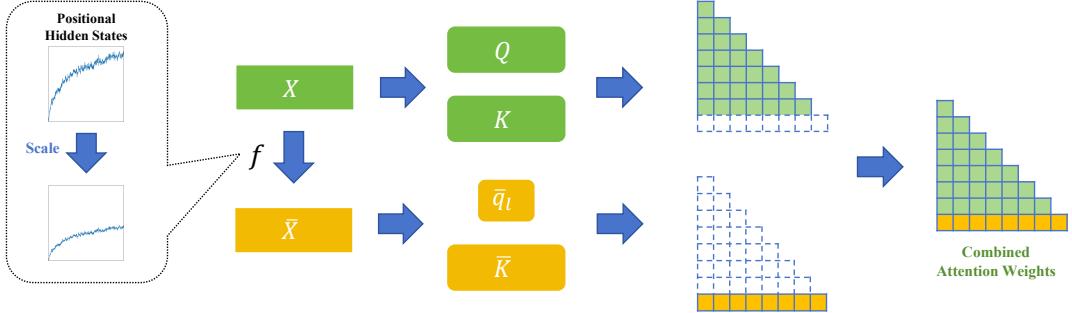


Figure 5: The framework of scaling positional hidden states and modifying attention.

likely to be retrieval-related layers), for example, the 10th to 25th layers with total 32 layers. The specific layer selection is also based on engineering experience.

4 Experiments

4.1 Setup

Evaluation Tasks and Models We apply our method to a wide range of popular open-source LLMs, including: 1) RoPE (Chen et al., 2023) models: LLaMA-2 (7B, 13B) (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023), Gemma-7B (Team et al., 2024), Qwen1.5-7B (Bai et al., 2023); 2) Context window extended models: Vicuna-16k (7B, 13B) (Chiang et al., 2023); 3) Alibi (Press et al., 2022) models: MPT-30B (Team, 2023). All the models are instruction-tuned versions.

And we evaluate the performance in 2 aspects: 1) Position-bias-related tests: “lost in the middle” benchmark, including NaturalQuestion multi-document QA (Liu et al., 2024) and KV retrieval (Liu et al., 2024) with the Gold document or KV at different positions in the context. The multi-document QA task includes 20 documents with a prompt length of about 2.3k tokens, while the KV retrieval task includes 140 KV pairs with an average length of about 10k tokens. 2) General long-context multi-task benchmark: LongBench (Bai et al., 2024), including multi-document QA, single-document QA, summarization, few-shot learning, synthetic tasks, and code completion, totaling 16 tasks with an average length of 37k tokens.

For prompts that exceed the context windows of LLMs, we follow LongBench’s approach by truncating from the middle and retaining the head and tail of the prompt to fit within the context windows. We use the metrics and scripts provided by the benchmark’s github repository for evaluation. More details about the benchmarks are in Appendix

B.

Implementation Details We implement our approach using PyTorch and HuggingFace Transformers in an A100 GPU. To ensure stable and reproducible results, we use greedy decoding in all generation experiments.

In the searching algorithm, we set the top- K to top-10 and ε to $L/4$, where L is the number of total layers. The search process takes approximately 10 minutes. The details of the scaling channels, layers, and factors are shown in Appendix C.3.

Baselines Besides the original models, we include another training-free positional bias mitigation method as the baseline, which is signified by **w/ Ms-PoE** (Zhang et al., 2024). It is a head-wise position embedding scaling method, and we follow its default settings to apply scaling coefficients of 1.2 to 1.8 starting from the 3rd layer.

4.2 Main Results

From the evaluation results of “lost in the middle” benchmark in Table 1, several conclusions can be drawn: 1) Our method better improves overall performance at most positions, with the average improvement of up to 9.3%, 15.2% in NQ and KV retrieval, respectively, except for LLaMA-2-13B in KV retrieval. 2) Our method effectively enhances LLMs’ exploitation of information located in the middle and rear parts of the prompt. When key information is at the beginning of the prompt, performance is sometimes increased or decreased. Considering only the average performance of the last four positions, our method’s improvements over the original increase to 11.3% and 16.8% in NQ and KV retrieval, respectively, much higher than MsPoE. 3) Our method demonstrates better generalization performance, showing improvement on nearly all types of models, regardless of RoPE models, context window extended

Methods	NaturalQuestion							KV Retrieval						
	1st	5th	10th	15th	20th	Avg.	0%	25%	50%	75%	100%	Avg.		
LLaMA-2-7b-chat	32.4	23.8	30.6	31.6	38.2	31.3	77.6	24.6	62.0	35.6	78.0	55.6		
LLaMA-2-7b-chat w/ Ms-PoE	40.8	29.2	33.0	32.8	39.6	35.1	95.0	29.8	21.4	51.8	89.8	57.6		
LLaMA-2-7b-chat w/ Ours	33.6	34.0	40.6	43.0	51.8	40.6	63.6	38.0	82.2	40.6	94.6	63.8		
LLaMA-2-13b-chat	45.2	39.6	40.4	44.2	51.0	44.1	74.2	39.0	70.4	84.4	86.8	71.0		
LLaMA-2-13b-chat w/ Ms-PoE	48.4	41.4	42.4	45.4	52.6	46.0	87.8	28.0	35.4	49.2	83.0	56.7		
LLaMA-2-13b-chat w/ Ours	50.6	43.4	45.0	49.4	58.2	49.3	41.2	17.0	49.6	76.8	84.8	53.9		
Vicuna-7b-v1.5-16k	70.4	54.8	46.8	45.8	47.8	53.1	98.4	0.8	0.2	0.2	0.2	20.0		
Vicuna-7b-v1.5-16k w/ Ms-PoE	67.0	55.2	50.6	46.8	48.2	53.6	97.4	36.8	15.6	5.2	6.6	32.3		
Vicuna-7b-v1.5-16k w/ Ours	63.8	57.6	53.6	51.2	55.6	56.4	95.4	22.0	12.6	5.2	20.4	31.1		
Vicuna-13b-v1.5-16k	67.4	48.2	45.2	45.6	44.4	50.2	95.6	74.2	64.2	58.8	18.2	62.2		
Vicuna-13b-v1.5-16k w/ Ms-PoE	70.0	51.4	46.8	42.8	47.0	51.6	91.8	59.4	71.6	74.4	48.8	69.2		
Vicuna-13b-v1.5-16k w/ Ours	67.4	51.4	47.6	48.8	48.0	52.7	97.2	83.4	80.8	68.8	35.4	73.1		
Mistral-7b-Instruct-v0.2	57.2	55.0	61.2	61.6	62.6	59.5	99.8	93.0	89.0	95.0	94.2	94.2		
Mistral-7b-Instruct-v0.2 w/ Ms-PoE	58.2	60.0	62.6	58.8	62.2	60.4	99.8	95.6	88.4	96.0	95.4	95.0		
Mistral-7b-Instruct-v0.2 w/ Ours	61.2	56.4	63.2	59.8	64.0	60.9	97.6	93.2	90.6	95.6	93.8	94.2		
Gemma-1.1-7b-it	29.6	25.2	28.2	29.6	27.4	28.0	98.6	67.0	62.4	83.4	100.0	82.3		
Gemma-1.1-7b-it w/ Ms-PoE	33.8	29.0	31.6	28.6	28.6	30.3	0.0	0.0	0.0	0.0	0.0	0.0		
Gemma-1.1-7b-it w/ Ours	35.4	31.4	36.0	35.4	35.0	34.6	97.6	95.8	97.6	96.8	99.6	97.5		
Qwen1.5-7b-chat	72.4	53.8	52.2	51.2	54.4	56.8	100.0	97.2	84.6	60.0	56.4	79.6		
Qwen1.5-7b-chat w/ Ms-PoE	67.4	49.8	48.2	47.4	47.0	52.0	3.4	1.4	2.8	2.6	0.6	2.2		
Qwen1.5-7b-chat w/ Ours	67.4	55.2	53.6	56.0	59.4	58.3	97.2	95.6	98.8	76.6	94.4	92.5		
MPT-30b-chat	75.6	49.6	39.0	33.4	39.6	47.4	71.4	34.8	31.6	41.6	74.0	50.7		
MPT-30b-chat w/ Ms-PoE	/	/	/	/	/	/	/	/	/	/	/	/		
MPT-30b-chat w/ Ours	75.0	48.8	41.6	40.6	44.0	50.0	99.0	65.8	48.6	46.6	69.4	65.9		

Table 1: Performance of different methods with different models on NaturalQuestions (20 docs) (Liu et al., 2024) and KV retrieval (140 KV pairs) (Liu et al., 2024) dataset. “/” signifies “not applicable”.

models like Vicuna-16K, or Alibi models like MPT. In contrast, Ms-PoE causes instability of the output of Qwen and Gemma in KV retrieval, leading to very low accuracy.

From the evaluation results of Longbench in Table 2, our method demonstrates varying degrees of improvement across different tasks, with the most significant increases being 1.5% in few-shot learning tasks, 3.4% in code tasks, 4% in synthetic tasks, 9.2% in single document QA tasks, and 1.9% in multi-document QA tasks. However, maybe because Longbench mainly focuses on comprehensiveness and reality, the influence of position bias on these tasks is relatively minimal, our method does not significantly improve the average scores (only a little). But it at least demonstrates that it will not impair the model’s original capability to handle various long context tasks.

4.3 Analysis

From Bias to Balance As shown in Table 1, there is an trend that our method generally increases performance when the key information is at the middle or end, but decreases when at the beginning. It reveals a possible fact that the po-

sitional hidden states may be an important factor causing the model to overly focus on the beginning parts of the context while miss the rear or middle parts. Therefore, scaling positional hidden states by a scaling factor less than 1 can reduce its impact, thus shift the model’s attention distribution from the bias to the beginning to a more balanced distribution.

In theory, a scaling factor over 1 amplifies the effect of positional hidden states, and one between 0 and 1 shrinks it. A negative factor can even reverse it. Correspondingly, as shown in Figure 6, a positive scaling factor over 1 causes the model to focus more on the beginning, while a negative factor shifts the focus towards the end. A factor between 0.5 and -1 leads to a relatively balanced attention distribution, where the average accuracy also peaks. These results demonstrate that scaling positional hidden states are essentially steering the position information in the model, to shift the position bias towards our desired direction.

Side Effects It is natural to concern scaling hidden states may impair the model’s normal workflow. So we utilized MMLU benchmark (Hendrycks et al., 2021), which assesses LLMs’

Models	SingleDoc	MultiDoc	Synth.	Summ.	FewShot	Code	AVG
LLaMA-2-7b-chat	28.9	29.7	6.6	26.3	61.2	47.1	33.3
LLaMA-2-7b-chat w/ Ms-PoE	29.8	31.7	10.5	26.7	61.0	48.1	34.6
LLaMA-2-7b-chat w/ Ours	29.2	29.3	9.7	25.0	61.6	46.9	33.6
LLaMA-2-13b-chat	21.4	14.6	11.2	26.1	61.5	39.8	29.1
LLaMA-2-13b-chat w/ Ms-PoE	20.8	15.4	12.7	27.3	62.8	36.3	29.2
LLaMA-2-13b-chat w/ Ours	30.6	9.6	10.8	25.7	62.6	43.2	30.4
Vicuna-7b-v1.5-16k	30.2	21.6	7.2	26.7	53.9	40.5	30.0
Vicuna-7b-v1.5-16k w/ Ms-PoE	32.3	24.2	8.3	28.0	55.2	43.1	31.8
Vicuna-7b-v1.5-16k w/ Ours	27.1	22.1	11.2	26.1	55.0	40.2	30.3
Vicuna-13b-v1.5-16k	31.1	33.8	21.2	26.2	62.0	39.8	35.7
Vicuna-13b-v1.5-16k w/ Ms-PoE	34.5	33.1	16.0	27.5	64.5	37.6	35.5
Vicuna-13b-v1.5-16k w/ Ours	30.1	35.1	25.0	25.8	63.5	41.7	36.9
Mistral-7b-Instruct-v0.2	37.8	28.5	49.7	28.8	65.3	52.9	43.8
Mistral-7b-Instruct-v0.2 w/ Ms-PoE	41.7	22.2	38.4	2.8	23.8	19.5	24.7
Mistral-7b-Instruct-v0.2 w/ Ours	38.4	30.4	49.8	29.4	64.8	52.9	44.3
Gemma-1.1-7b-it	39.4	23.2	32.2	24.2	14.4	19.8	25.5
Gemma-1.1-7b-it w/ Ms-PoE	41.7	22.2	38.4	24.9	14.0	19.5	26.8
Gemma-1.1-7b-it w/ Ours	39.0	23.0	35.5	24.5	14.9	19.3	25.7
Qwen1.5-7b-chat	46.4	39.5	38.4	22.3	56.4	50.2	42.2
Qwen1.5-7b-chat w/ Ms-PoE	42.0	41.5	30.3	25.7	46.5	38.0	37.3
Qwen1.5-7b-chat w/ Ours	45.8	38.8	38.5	22.1	57.6	49.6	42.2
MPT-30b-chat	27.9	21.9	7.5	25.7	57.3	39.3	29.9
MPT-30b-chat w/ Ms-PoE	/	/	/	/	/	/	/
MPT-30b-chat w/ Ours	29.4	19.5	6.7	25.8	57.6	40.1	29.9

Table 2: Performance of different methods with different models on LongBench (Bai et al., 2024).

Model	MMLU	Reorder
Vicuna-7b-v1.5-16k	48.22	20.83
Vicuna-7b-v1.5-16k w/ Ours	48.38	20.83
Qwen1.5-7b-chat	60.84	28.13
Qwen1.5-7b-chat w/ Ours	61.43	28.13
Mistral-7B-Instruct-v0.2	60.31	18.75
Mistral-7B-Instruct-v0.2 w/ Ours	60.38	19.79

Table 3: Performance of difference models on MMLU and the timeline reorder task, before and after applying our method.

general capabilities, and the timeline-reorder task from LooGLE (Li et al., 2024), which requires arranging events chronologically across extensive text and is sensitive to positional information, to evaluate whether our method will impair the original abilities of LLMs. As shown in Table 3, there is no significant detriment to the models’ performance with our methods in default settings.

5 Conclusion

We propose a method which scales positional hidden states to mitigate the position bias issue of LLMs. Specifically, our study first explore the re-

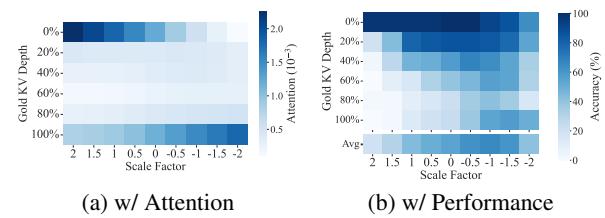


Figure 6: Attention distribution on KV pairs at different positions (depths) and performance when scaling the 2,393rd channel of Vicuna-7b with different scale factors on KV retrieval (Liu et al., 2024) of 100 KV pairs.

relationships between position bias (in the model’s performance), attention weights, causal mask, positional information in hidden states and positional channels, to confirm the positional channels can affect position bias. Based on these findings, we design a positional channel search algorithm to identify positional hidden states, and mitigate the model’s position bias by scaling the positional hidden states. These findings provide a new perspective for research related to position information and position bias of LLMs.

Limitations

The selection of layers to apply is mainly based on engineering experience. And although our method has tried to minimize the side effect of scaling hidden states, if choosing a too large scale factor or an inappropriate channel to scale, we still observe significant performance degradation.

Some newly released models like Qwen2.5 (Qwen et al., 2024) have achieved nearly perfect performance on retrieval-based long-context tasks. Thus, on traditional benchmarks, our method may not bring more improvement. Experiment on more newly released models and benchmarks may still be needed in the future.

References

- Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emmann Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojtaba Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Ziqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sam-budha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyra Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). Preprint, arXiv:2404.14219.
- Shengnan An, Zexiong Ma, Ziqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024. [Make your LLM fully utilize the context](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. [Qwen technical report](#). ArXiv preprint, abs/2309.16609.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. [LongBench: A bilingual, multi-task benchmark for long context understanding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.
- Avi Caciularu, Matthew Peters, Jacob Goldberger, Ido Dagan, and Arman Cohan. 2023. [Peek across: Improving multi-document modeling via cross-document question-answering](#). In *The 61st Annual Meeting Of The Association For Computational Linguistics*.
- Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. [Extending Context Window of Large Language Models via Positional Interpolation](#). ArXiv preprint, abs/2306.15595.
- Yuhan Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and Rui Yan. 2024. [Fortify the shortest stave in attention: Enhancing context awareness of large language models for effective tool use](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11160–11174.
- Ta-Chung Chi, Ting-Han Fan, Li-Wei Chen, Alexander Rudnicky, and Peter Ramadge. 2023. [Latent positional information is in the self-attention variance of transformer language models without positional embeddings](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1183–1193.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#).
- Tri Dao. 2024. [Flashattention-2: Faster attention with better parallelism and work partitioning](#). In *The Twelfth International Conference on Learning Representations*.
- DeepSeek-AI. 2024. [Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model](#). Preprint, arXiv:2405.04434.
- Yifei Gao, Lei Wang, Jun Fang, Longhua Hu, and Jun Cheng. 2023. [Empower Your Model with Longer](#)

- and Better Context Comprehension. *ArXiv preprint*, abs/2307.13365.
- Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, LiuYiBo LiuYiBo, Qiangosun Qiangosun, Yuxin Liang, Hao Wang, Enming Zhang, and Jiaxing Zhang. 2024a. Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13628–13642, Bangkok, Thailand. Association for Computational Linguistics.
- Zhiyuan He, Huiqiang Jiang, Zilong Wang, Yuqing Yang, Luna K. Qiu, and Lili Qiu. 2024b. Position engineering: Boosting large language models through positional information manipulation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7333–7345, Miami, Florida, USA. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillem Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *ArXiv preprint*, abs/2310.06825.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.
- Greg Kamradt. 2023. Needle in a haystack - pressure testing llms.
- Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024. LooGLE: Can long-context language models understand long contexts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16304–16333, Bangkok, Thailand. Association for Computational Linguistics.
- Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhui Chen. 2025. Long-context LLMs struggle with long in-context learning. *Transactions on Machine Learning Research*.
- Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2025. World model on million-length video and language with blockwise ringattention. In *The Thirteenth International Conference on Learning Representations*.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Alexander Peysakhovich and Adam Lerer. 2023. Attention Sorting Combats Recency Bias In Long Context Language Models. *ArXiv preprint*, abs/2310.01427.
- Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Bin Yuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittweiser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv preprint*, abs/2403.05530.
- Freida Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärlí, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context.
- 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. In *First Conference on Language Modeling*.
- Ruixiang Tang, Dehan Kong, Longtao Huang, and Hui Xue. 2023. Large language models can be lazy learners: Analyze shortcuts in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4645–4657.
- 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*, abs/2403.08295.
- MosaicML NLP Team. 2023. [Introducing mpt-30b: Raising the bar for open-source foundation models](#). Accessed: 2023-06-22.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *ArXiv preprint*, abs/2307.09288.
- Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. 2024. [Length generalization of causal transformers without position encoding](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14024–14040, Bangkok, Thailand. Association for Computational Linguistics.
- Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham M. Kakade, Hao Peng, and Heng Ji. 2025. [Eliminating position bias of language models: A mechanistic approach](#). In *The Thirteenth International Conference on Learning Representations*.
- Wenhai Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2025. [Retrieval head mechanistically explains long-context factuality](#). In *The Thirteenth International Conference on Learning Representations*.
- Qingyu Yin, Xuzheng He, Xiang Zhuang, Yu Zhao, Jianhua Yao, Xiaoyu Shen, and Qiang Zhang. 2024. [Stablemask: Refining causal masking in decoder-only transformer](#). In *Forty-first International Conference on Machine Learning*.
- Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. [Yi: Open foundation models by 01.ai](#). *ArXiv preprint*, abs/2403.04652.
- Yijong Yu. 2023. [Training With "Paraphrasing the Original Text" Improves Long-Context Performance](#). *ArXiv preprint*, abs/2312.11193.
- Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. 2024. [Found in the middle: How language models use long contexts better via plug-and-play positional encoding](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- ## A Related Works
- Addressing Position Bias** Recent works reveal that LLMs often exhibit position bias, also known as the "lost in the middle" phenomenon (Liu et al., 2024; Kamradt, 2023). Previous efforts to mitigate this bias fall into several categories: 1) RoPE-based methods: These approaches modify the RoPE computation process to alleviate long-distance attention decay (less attention means less information retrieved), including Attention Bucket (Chen et al., 2024), which uses an ensemble of multiple RoPE bases to mitigate position bias, and Ms-PoE (Zhang et al., 2024), which dynamically interpolates with a small coefficient for different heads. 2) SFT-based methods (He et al., 2024a; Yu, 2023; An et al., 2024): These methods construct data with more diverse key information distributions or employ system2think SFT tasks to mitigate position bias. They require further training of the model. 3) Attention mask-based methods (He et al., 2024b): These methods modify attention mechanisms, including Attention Transition (Gao et al., 2023), which redirects attention to significant parts of the context and Stable Mask (Yin et al., 2024), which introduces pseudo attention into the causal mask, ensuring stable attention distribution when facing lengthy texts. 4) Prompt-based methods (Jiang et al., 2024; Peysakhovich and Lerer, 2023): These methods introduce an external module to reorder or compress information in the prompt, thereby mitigating position bias.

LLMs. It contains six major categories, covering single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks and code completion. The evaluation metrics are: F1 for single-document QA and multi-document QA, Rouge-L for summarization, accuracy (exact match) for few-shot learning and synthetic tasks, and edit similarity for code completion. During inference, since the original context may sometimes be too long, the input sequences will be truncated in the middle part to avoid exceeding the context window of the model.

C Other Implementation Details

C.1 Curve Fitting

When we perform curve fitting on $h(p)$, we use least-squares cubic polynomial fit. And when judging its monotonicity, we skip the first 30 positions because the first a few values are often outliers. Since $h(p)$ is originally a discrete function, in practice, we employ the second-order difference to approximate the second-order derivative when computing smoothness.

C.2 Ms-PoE on Mistral

When applying Ms-PoE (Zhang et al., 2024) to mistral-7b (Jiang et al., 2023) with its default parameters (minimal scale factor is 1.2 and maximal is 1.8), we found the model fail to generate normal responses, so we set the maximal scale factor to 1.2, under which Ms-PoE (Zhang et al., 2024) is equal to PI (Chen et al., 2023) with scale factor 1.2.

C.3 Parameters of the Scaling Method

Model	Channel Index	Scale factor	Applied layers
LLaMA-2-7b-chat	2,393	-1	10~25
LLaMA-2-13b-chat	4,283	-1	10~34
Vicuna-7b-v1.5-16k	2,393	0	10~25
Vicuna-13b-v1.5-16k	4,923	0	10~34
Mistral-7B-Instruct-v0.2	213	0	10~25
Gemma-1.1-7b-it	1,665	0	10~22
Qwen1.5-7b-chat	1,081	0.2	10~25
MPT-30b-chat	6,926	0	10~42

Table 4: The scaled channels, scale factors and applied layers of models.

The scaled channel indices, scale factors and applied layers of each model we use in our experiments are shown in Table 4.

C.4 Inference Latency

Table 5 shows the running time of LLaMA-2-7b-chat with different methods in the KV retrieval

Method	KV Retrieval	Multi-Doc
FlashAttention-2	22	14
Ours	32	15
Ms-PoE	61	26

Table 5: Average inference time (minutes) of LLaMA-2-7b-chat in a single A100 on KV retrieval.

dataset consisting of 500 samples with average length of about 10,000, and the multi-document QA dataset consisting of 500 samples with average length of about 3,300. Our method requires recompute the query and key states, thus inevitably requires more time compared to baseline, but the cost is within an acceptable range. In contrast, Ms-PoE (Zhang et al., 2024) need to compute the attention weights twice, resulting in a doubling of time consumption.

D Details of The Confirmatory Experiments

D.1 Obtain Attention to Key Information

To avoid the influence of internal knowledge in the model and make attention calculation simpler, we conduct a KV retrieval task, whose prompt format is as follows:

Json data:
{ "os08jbk1limft6wgcxeda":
"imx6lyp4b8ogjaq7ret1",
.....(n key-value pairs)}

The value of key "os08jbk1limft6wgcxeda" is "

The last token of the prompt will directly account for predicting the answer, i.e., the value which need to be retrieved. Hence, the last token’s attention weights to the previous text can reflect whether it accurately retrieves the key information. We define the model’s attention (in some layer) to the key information as A_G in Eq 4, where G represents the set of token positions corresponding to where the key information is at, l is the position of the last token of the prompt, and $a_{l,j}$ represents the attention weight of the l -th token to the j -th token. By shifting G , we use the same method to calculate its attention to each other KV pairs.

$$A_G = \frac{1}{|G|} \sum_{j \in G} a_{l,j} \quad (4)$$

D.2 Attention is Related to Performance

As shown in Figure 7, when we manually double the attention weights of the gold KV (i.e. the 25th KV, as illustrated in Figure 7b) during the model’s forward pass on the KV retrieval task, the retrieval accuracy when the gold KV is the 25th KV improves, (shown in Figure 7d). This demonstrates that the attention weights to the key information are positively correlated with retrieval accuracy.

D.3 How We Modify Causal Mask and Position Embedding in KV Retrieval

In the method 1 in section 2.2, we crop the causal mask to let the “key tokens” unable to attend the previous tokens. As shown in Figure 8, the white part represents the cropped part, which means attention weights are 0, and the orange part represents the attention between tokens within key tokens. In addition, we have retained the attention of key tokens to the first token to maintain the stability of attention distribution. What is more, we only modify the causal mask in layers 1~8, but as the results, the attention to the key information is still significantly improved in layers 15~31, which indicates the positional information generated by causal mask in former layers can be transmitted to latter layers using posisional hidden states as the medium, thus modifying the causal mask solely in the former layers can induce a profound shift in the model’s comprehension of positional information.

In the method 2 and 3 in section 2.2, we modify the position embeddings through altering the position ids. The specific operation is shown in the Figure 9, in which we directly replace the position ids corresponding to the key tokens with the position ids of the starting tokens (or the ending tokens), and actually only the attention weights of the last token to previous tokens are modified. We apply this modification in all the layers. Compared to modifying the causal mask, if only modify position embedding in former layers, the attention in the latter layers remains almost unchanged, which indicates the positional information generated by position embedding may be temporary and can hardly be transmitted across layers.

D.4 Perturbation on Causal Mask and Position Embedding

To further explore the origin of these position hidden states, we performed perturbation experiments. As depicted in Figure 11c, subtracting 200 from the

position ids corresponding to the 400th to 600th tokens (reducing PE) had only a minor effect on the position hidden states, whereas, in Figure 11b, crop the causal mask to make the 400th to 600th tokens unable to attend the 1st to 400th tokens (cropping causal mask) led to significant fluctuations in positional hidden states of the 400th to 600th tokens. This result proves the causal mask is the main factor causing this kind of positional hidden states, and it is the token’s position in the causal mask that determines its value in the positional hidden states, but not position ids of position embedding.

D.5 Positional Hidden States Affect Attention

Although it is intuitive that these monotonically changing hidden states channels convey rich positional information, it is unclear how much impact one position channel will have on position bias. To find out this, we modify the position hidden states values to see corresponding attention changes, in a KV retrieval task with 50 KV pairs while the gold KV is the 26th KV.

Specifically, we subtract 0.3 from the 213rd channel of the hidden states of the tokens of the 26th and 36th KV respectively (in layer 15 to 20 of Mistral-7b-v0.2), and observe the change of positional hidden states (the 213rd channel) and attention weights of each KV.

As shown in Figure 10, we show each KV’s average hidden state value of the 213rd channel of Mistral-7b-v0.2 and the average attention to each KV, which are both averaged across layer 15 to 20 (because they are retrieval-related layers). It is clear that when the positional hidden states values of the 26th or 36th KV greatly drop, the attention to it is greatly increased, even over the level of the 1st KV, regardless of whether it is the gold KV. Therefore, we confirm only changing one positional channel can significantly affect attention, thus affect position bias.

E Ablation Experiments of the Searching Algorithm

To evaluate the contributions of different components in our method, we introduce the following sets for the ablation study: (1) Ours w/o monotonicity, w/o smoothness, and w/o validation set, which adjust the search algorithm by not considering one of these three indicators, respectively (details in Appendix C). (2) Ours w/ scale 2 channels, which scales the top-2 positional hidden states simultane-

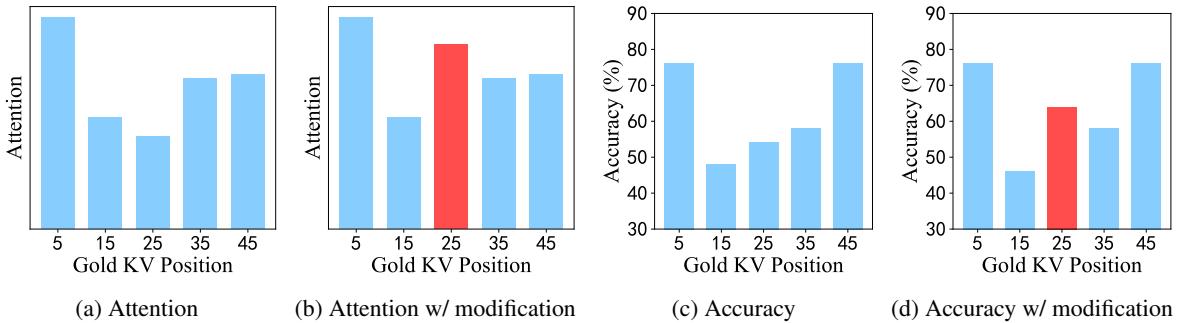


Figure 7: Distribution of attention weight and accuracy as the gold KV is placed at different positions in the prompt. (b) and (d) are situations when the attention on the 25th KV pair is manually modified.

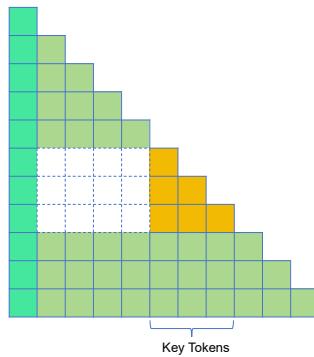


Figure 8: Cropping the causal mask to let key tokens unable to see previous tokens, except the first token.

ously. (3) Ours w/ modify last 16 tokens and w/ modify all tokens, which adjust the range of tokens affected by the scaling operation in Equ.(3).

Table 6 shows the ablation results. It can be seen that without filtering by monotonicity or smoothness, performance may decline, and removing the validation set results in more decline in model performance. When the range of tokens or channels affected by scaling is expanded, most models experience varying degrees of performance loss. Considering these factors, we choose to modify only the last token and the top-1 positional channel to achieve the best performance.

F Attention Distribution Layer-wise and Head-wise

Figure 12 shows Mistral-7b’s attention to each KV pair of each layer (average across all attention heads) in the context in a KV retrieval task when the gold KV is put at different positions. The y-axis is the gold KV’s position, x-axis is each KV’s position, and the scale of the colorbar represents attention (10^{-3}). We can observe that diagonal patterns, which indicates the attention is concentrated on the “key tokens”, appear only in the latter layers

(start from layer 14), and may be a manifestation of retrieval behavior. In contrast, the former layers only focus on the beginning or end, regardless of where the key information is located.

Figure 13 shows the head-wise situation of layer 15. We can see actually only a portion of attention heads exhibit diagonal patterns, which may correspond to *retrieval heads* (Wu et al., 2025). The attention distribution in these heads also shows a pattern corresponding “lost in the middle”, being larger at the beginning or end while significantly smaller at the middle.

G Positional Hidden States Visualization

We show various models’ positional hidden states of each layer in Figure 14. When visualizing, we discard the first 30 tokens because the hidden states values of these tokens are often too huge (usually hundreds of times larger than the normal value (Sun et al., 2024)), which can disrupt monotonicity. We observe its monotonic trend may first appears just in the first layer (actually just after the first attention module), and continues to be more marked.

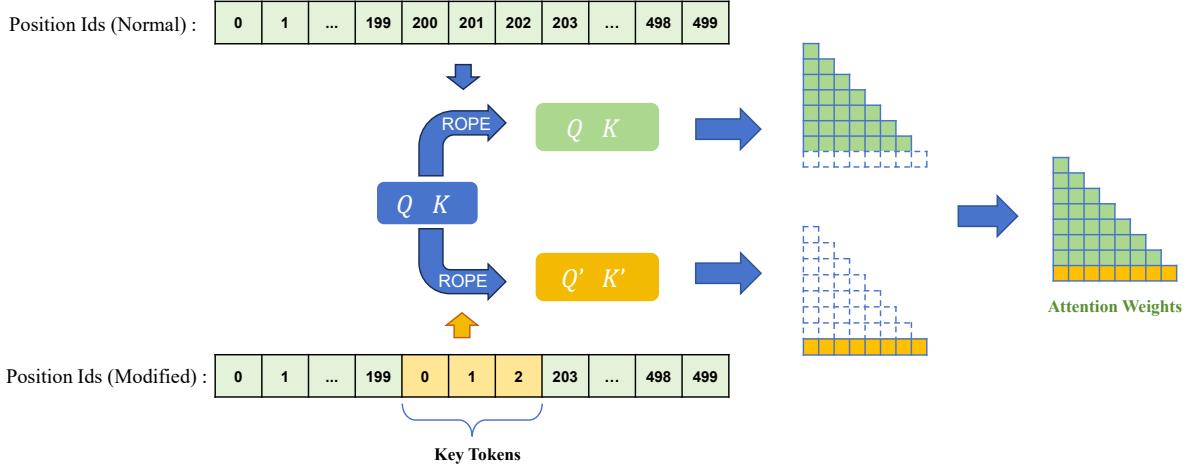


Figure 9: Shifting position ids to the start (PE to beginning).

Method	LLaMA-2-7b	Vicuna-13b	Gemma-7b	Mistral-7b	Qwn1.5-7b
not applied	31.3	50.2	28.0	59.5	56.8
Ours	40.6	52.7	34.6	60.9	58.3
w/o monotonicity	40.6	51.8	34.6	60.9	58.3
w/o smoothness	40.6	52.7	27.8	60.9	58.3
w/o validation set	30.1	51.8	26.5	60.9	58.3
w/ scale top-2 channels	37.2	50.8	31.7	60.1	57.2
w/ modify last 16 tokens	41.6	51.5	34.6	59.7	58.1
w/ modify all tokens	44.0	50.8	31.7	59.5	57.4

Table 6: Average performance of different ground-truth positions using different methods on NaturalQuestions multi-document QA dataset (20 documents) (Liu et al., 2024).

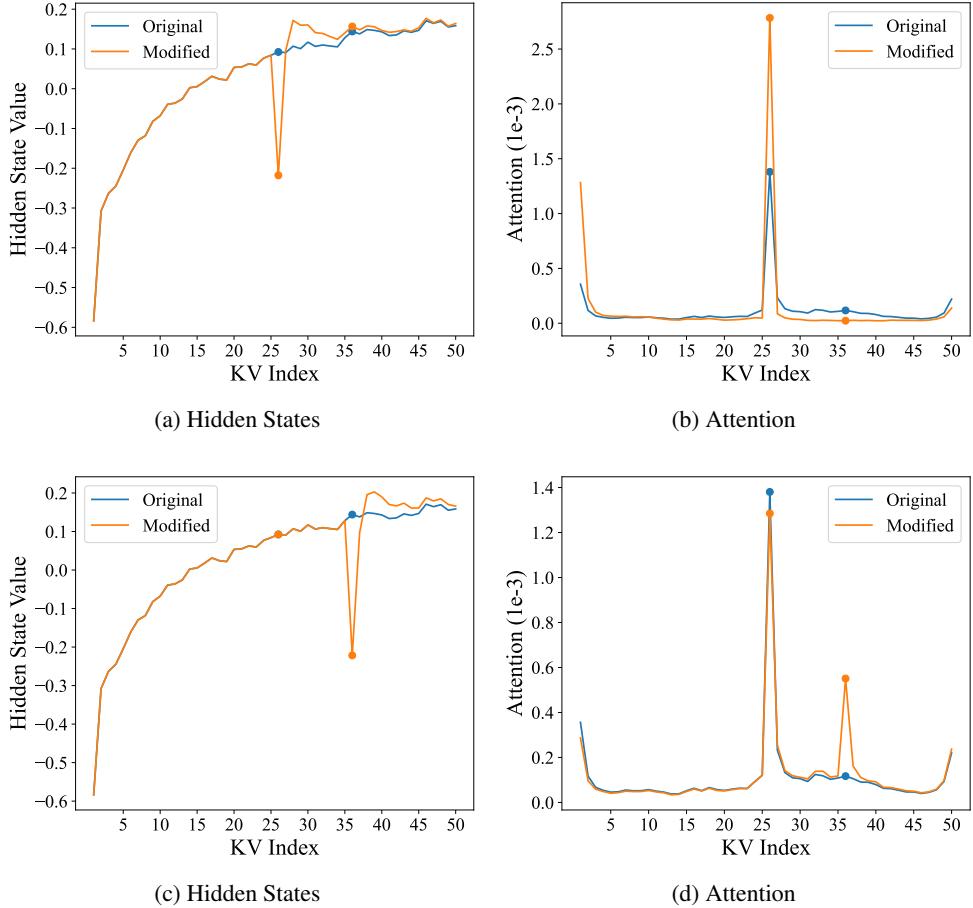


Figure 10: The positional hidden states and attention of each KV in a KV retrieval task (the gold KV is the 26th KV) of Mistral-7b-v0.2. (a)(b) We modify the hidden states of the 26th KV. (c)(d) We modify the hidden states of the 36th KV. (a)(c) The value of the 213rd channel of the hidden states (averaged across layer 15 to 20) of each KV (averaged across the tokens of the KV). (b)(d) The attention (averaged across layer 15 to 20 and across every attention head) to each KV.

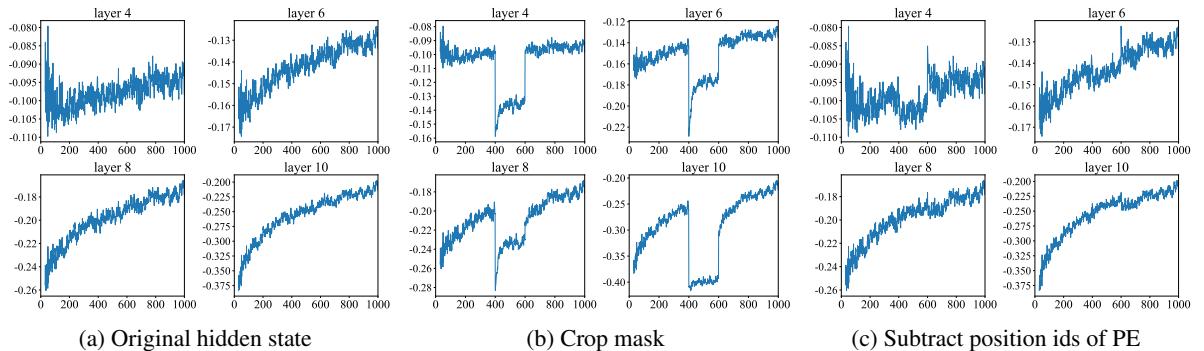


Figure 11: We performed perturbation experiments on the causal mask and position embedding (PE), showing the 213rd channel of hidden states of some layers of Mistral-7b (Jiang et al., 2023) using randomly synthesized corpus as input. It is clear that only cropping mask cause significant drop of positional hidden states values.

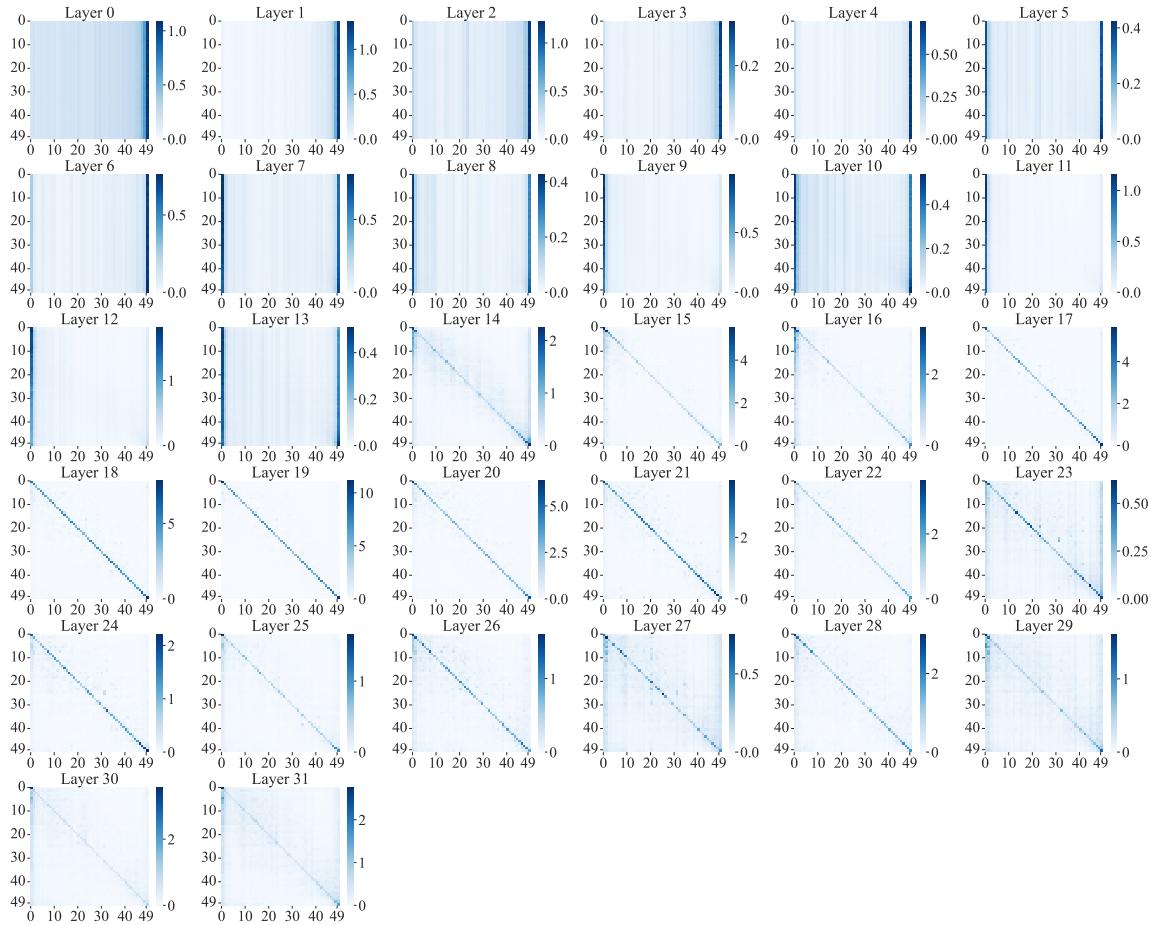


Figure 12: The average attention weight distributed on each KV, of all the 32 layers of Mistral-7b, on a 50 KV pairs retrieval task, when the gold KV is put at each different position.

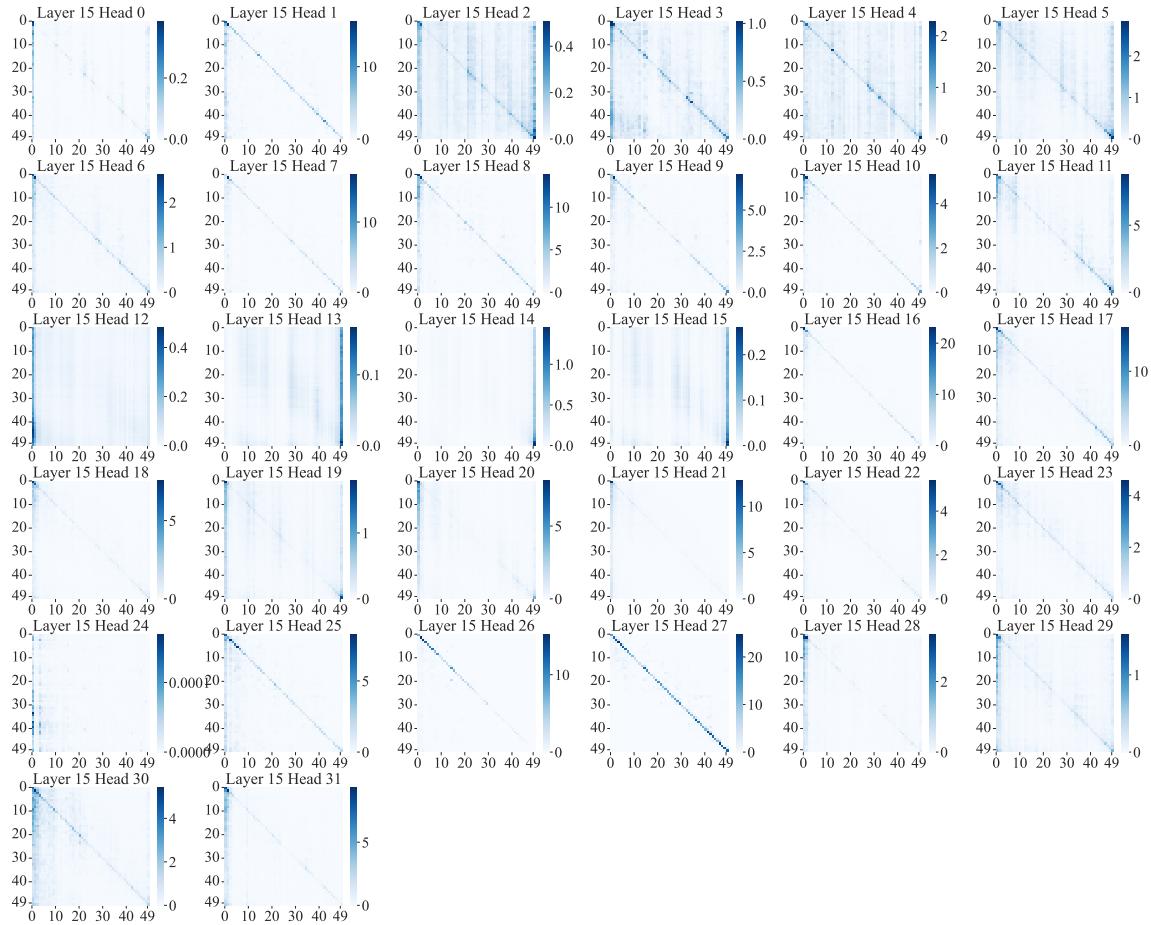
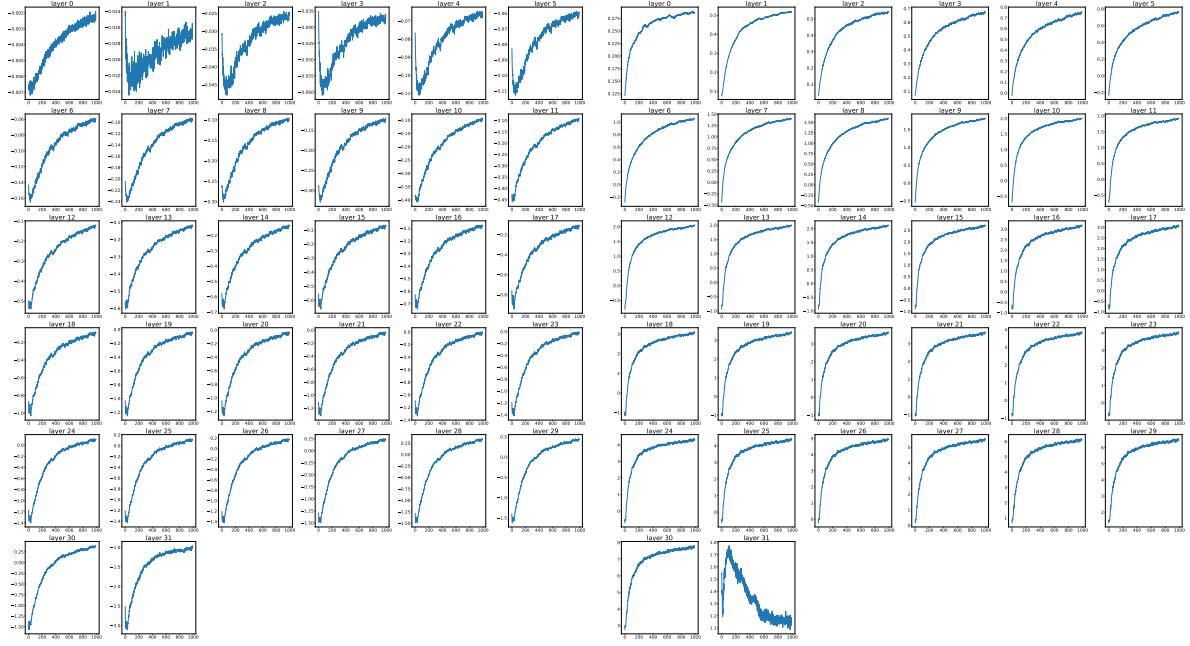
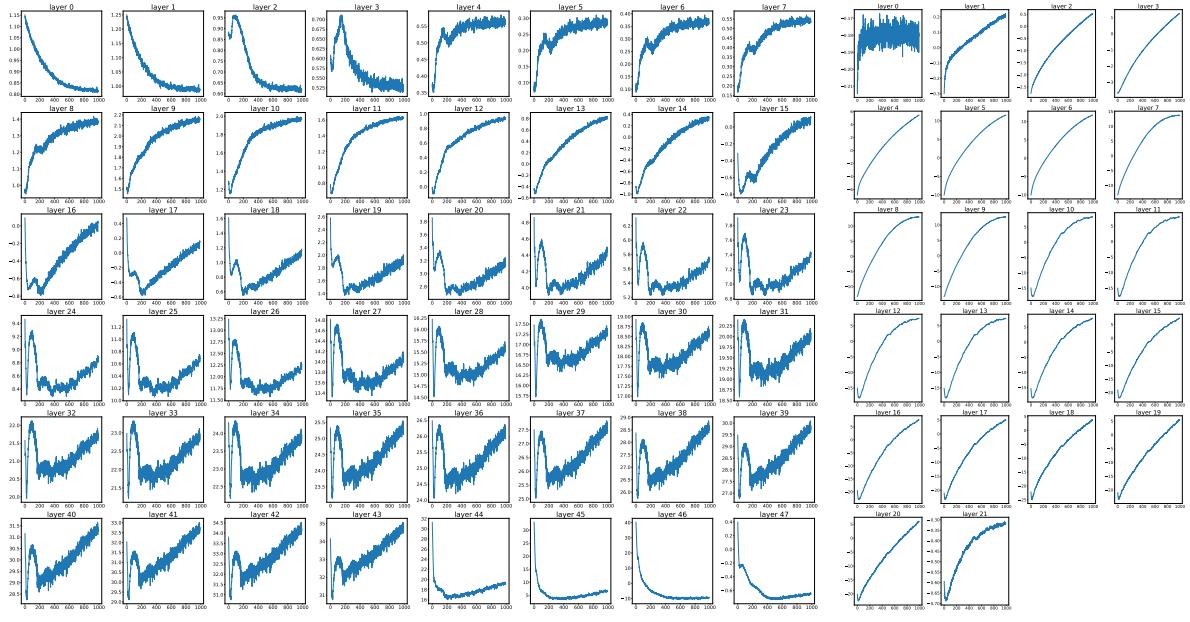


Figure 13: The average attention weight distributed on each KV, of all the 32 attention heads of layer 15 of Mistral-7b, on a 50 KV pairs retrieval task, when the gold KV is put at each different position.



(a) mistral-7b

(b) LLaMA-2-7b



(c) MPT-30b

(d) Tinyllama-NoPE

Figure 14: Hidden states values with the token positions of the positional channel of each layer. The x-axis represents the position, and the y-axis represents the value of the states. (a) The 213rd channel of Mistral-7b-v0.2 (b) The 2393th channel of Llama2-7b (c) The 1942th channel of MPT-30b (d) The 1156th channel of TinyLlama-NoPE-1.1B, which is a model without any position embeddings.