PIC: Unlocking Long-Form Text Generation Capabilities of Large Language Models via Position ID Compression

Haoran Que[1], **Wenge Rong**[2]
[1]Sino-French Engineer School, Beihang University, China
[2]School of Computer Science and Engineering, Beihang University, China
2224124@buaa.edu.cn, w.rong@buaa.edu.cn

Abstract

Long-context understanding is crucial for large language models (LLMs) and has become a fundamental capability for most LLMs. However, beyond the focus on “input-long”, the ability to “output-long” is equally significant, yet it remains underexplored. To address this limitation, we propose a simple, efficient, and plug-in approach, **Position ID Compression (PIC)**, to unlock the long-form text generation potential of LLMs. The idea is straightforward: by compressing the position ids of the context, we provoke and guide LLMs to generate coherent and longer output. Specifically, we find that directly reducing the position ids by a fixed ratio significantly impacts the generation quality. To mitigate this, we propose two variants of PIC: **NTK-aware PIC** and **Dynamic PIC**. Without additional training, both methods enable LLMs to extend their generation length by approximately 1.5 times without compromising generation quality. Furthermore, by integrating supervised fine-tuning (**SFT**) with PIC, we propose **PIC-SFT**, which further improves LLMs' long-form text generation capabilities, achieving top performance on HelloBench and LongBench-Write. Extensive experiments demonstrate the effectiveness of our approach.

1 Introduction

Modeling long context is essential for Large Language Models (LLMs) as it meets the user's need for long-range interactions (Ding et al., 2024; Lin et al., 2024) while enhancing the capabilities of LLM-based systems (e.g., Retrieval-Augmented Generation(Gao et al., 2023), Multi-Agent System(Li et al., 2024), etc.). Recently, a variety of methods for context window extension(Chen et al., 2023; Peng et al., 2023) and efficient inference (e.g., sparse attention(Beltagy et al., 2020; Xiao et al., 2023), KV cache compression(Dao, 2023; Kwon et al., 2023)) have been proposed. As a result, long-context modeling has become a fundamental capability of current LLMs(Dubey et al., 2024; Yang et al., 2024), with some models even capable of handling inputs with millions of tokens(Team et al., 2023; Zeng et al., 2022). Beyond long-context understanding, long-context modeling is also related to long-form text generation. With an extended context window, a model should theoretically be capable of both “input-long” and “output-long”. However, this is often not the case in practice: while models can process millions of tokens as input, they struggle to generate even 4000 tokens(Que et al., 2024). This motivates us: *Can we efficiently unlock the long-form text generation capabilities of LLMs based on existing long-context modeling paradigms?*

In this work, we propose **Position ID Compression (PIC)**, a simple, efficient, and plug-in method to unlock the long-form text generation potential of LLMs. As illustrated in Figure 1, the core idea of PIC is to directly or indirectly compress the position ids, so that the relative positions perceived by the model are smaller than the actual relative positions, thereby extending the model's generation length. To verify this idea, we first attempt to reduce the position ids by a fixed ratio, but we find that this method performs poorly. To address this, we propose two variants of PIC: **NTK-aware PIC** and **Dynamic PIC**. NTK-aware PIC indirectly reduces position ids by scaling down the rotation angle of Rotary Position Embedding (**RoPE**), while Dynamic PIC dynamically compresses position ids in the middle part of the context. Experimental results show that NTK-aware PIC and Dynamic PIC are effective and efficient. They can extend the model's generation length by approximately 1.5 times without additional training and demonstrate generalization capabilities.

Furthermore, we combine supervised fine-tuning (**SFT**) with PIC and propose **PIC-SFT**, significantly extending the generation length to 5 times the original without losing the generation quality.

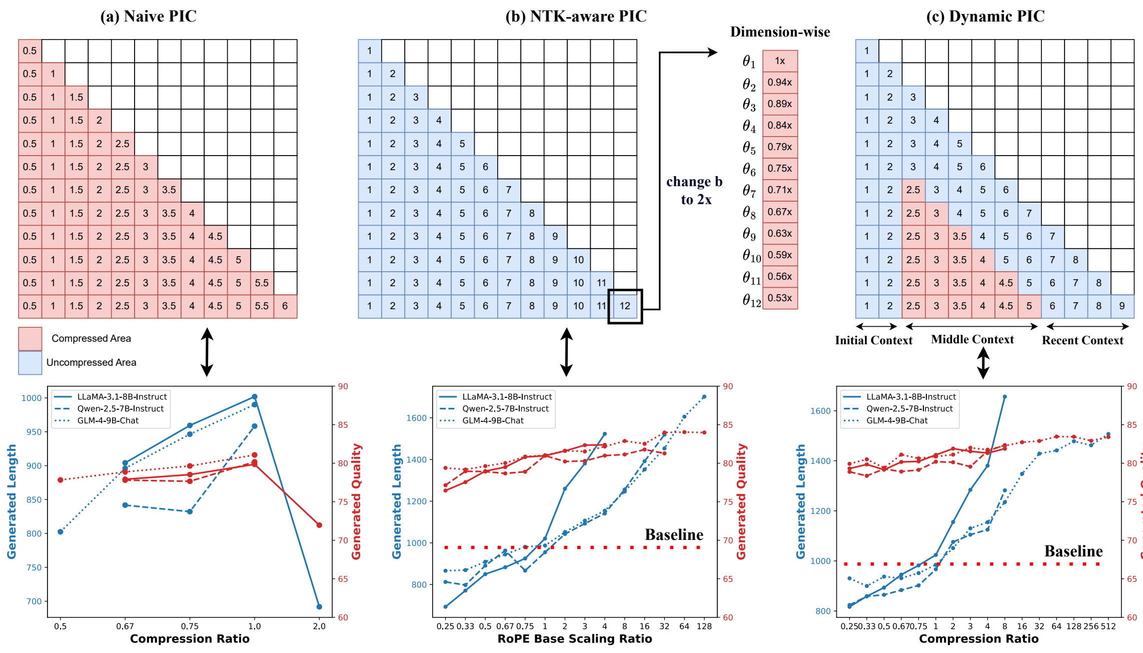


Figure 1: Illustration of Position ID Compression (PIC). (*Top*): The diagrams sequentially show the mechanisms of **Naive PIC**, **NTK-aware PIC**, and **Dynamic PIC**. The red areas represent the compressed areas, while the blue areas represent the uncompressed areas. The number on the block indicates the position id. (*Bottom*): The corresponding experimental results for the three PIC methods, where the blue lines represent the generated length and the red lines represent the generated quality.

Experimental results show that compared to traditional SFT, PIC-SFT achieves better performance on HelloBench (Que et al., 2024) and LongBench-Write (Bai et al., 2024). Moreover, through Naive Inference, PIC-SFT better preserves the short-form text generation capabilities of LLMs.

Our main contributions are as follows:

1. **PIC**: We introduce PIC, a method to extend the long-form text generation capabilities of LLMs by compressing position ids, allowing for longer generation length without additional training.
2. **NTK-aware PIC and Dynamic PIC**: We propose two variants, NTK-aware PIC and Dynamic PIC, to improve the efficiency and effectiveness of PIC.
3. **PIC-SFT**: We combine PIC with SFT, resulting in PIC-SFT, which extends the generation length up to 5 times the original while preserving short generation capabilities.

2 Background and Related Work

2.1 RoPE

RoPE was first introduced in RoFormer (Su et al., 2024) and has been widely used in current LLMs (Touvron et al., 2023; Bai et al., 2023a). For a sequence of vectors $(x_1, x_2, \dots, x_i, \dots, x_n)$ where $x_i \in \mathbb{R}^d$ represents the word embedding of the i-th token in the sequence. RoPE aims to incorporate relative positional information into the computation of attention scores. Specifically, it ensures that the inner product of the query q_m and key k_n encodes position information in the relative form:

$$\langle f_q(x_m, m), f_k(x_n, n) \rangle = g(x_m, x_n, m-n). \quad (1)$$

m and n represent the position of the query and the key. To solve the functions f_q, f_k, and g, RoPE applies a rotary matrix to q_m and k_n:

$$\begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix},$$

where $\theta_j = b^{-2(j-1)/d}, j \in \{1, 2, ..., d/2\}$ and b is a fixed value called **RoPE Base**.

2.2 Position Interpolation

RoPE encodes a word embedding x_m by:

$$\{q_m, k_m\} = f_{\{q,k\}}(x_m, g(m), h(\theta_j)) \quad (2)$$

$$g(m) = m, \quad h(\theta_j) = b^{-2(j-1)/d}. \quad (3)$$

Based on RoPE, to further extend the context window of LLMs, Position Interpolation (**PI**) (Chen et al., 2023) alters $g(m)$:

$$g(m) = \frac{m}{s}, \quad \text{where} \quad s = \frac{L_t}{L_c}, \quad (4)$$

where s is the scaling factor, L_t is the size of the extended context window and L_c is the current context window size. Compared to Position Extrapolation, PI is more stable and requires only a small amount of training data to extend the context window.

2.3 NTK-aware Interpolation

Besides scaling all dimensions equally, there is a class of interpolation methods based on Neural Tangent Kernel (NTK) theory (Jacot et al., 2018). We focus on two widely used methods: **NTK-aware** (bloc97, 2023b) and **NTK-by-parts** (bloc97, 2023a). NTK-aware modifies $h(\theta)$ of the RoPE as follows:

$$h(\theta_j) = \left(b \cdot s^{\frac{d}{d-2}}\right)^{-2(j-1)/d}. \quad (5)$$

The core idea of NTK-by-parts is to interpolate for lower frequency dimensions while keeping higher frequency dimensions unchanged:

$$h(\theta_j) = (1 - \gamma(r))\,\frac{\theta_j}{s} + \gamma(r)\theta_j, \quad (6)$$

where γ is a step function that depends on r and r represents the frequency of the dimension.

2.4 Related Work

Long-Context Understanding Long-context understanding focuses on enabling LLMs to handle, process, and retrieve information from very long inputs (Hsieh et al., 2024; Bai et al., 2023b). To achieve input lengths of even 100M tokens (Team et al., 2024), various methodologies have been proposed (Shen et al., 2021; Ainslie et al., 2023). Among them, speed and length are the key factors in long-context modeling. Context window extension (Cobbe et al., 2021; Zhu et al., 2023; Jin et al., 2024) focuses on expanding the context window of LLMs to support a larger number of input tokens. KV cache compression (Liu et al., 2024b; Ge et al., 2023) aims to compress the KV cache, enabling faster inference and eliminating redundant information.

Long-Form Text Generation Long-form text generation is a crucial capability for LLMs, closely tied to their practical applications in real-world scenarios (Guan et al., 2022; Hosseini et al., 2024; Wei et al., 2024). To explore the long-form text generation capabilities of LLMs, many benchmarks and methods have been proposed (Ye et al., 2025; Tan et al., 2024; Liu et al., 2024a). Bai et al. propose LongBench-Write (Bai et al., 2024), a comprehensive benchmark for evaluating ultra-long generation capabilities. Que et al. propose HelloBench (Que et al., 2024), a comprehensive, in-the-wild, and open-ended benchmark to evaluate LLMs' performance in generating long text. Self-Lengthen (Quan et al., 2024) leverages the intrinsic knowledge and skills of LLMs to enable the model to generate longer content. Suri (Pham et al., 2024) and LongWriter construct high-quality datasets to train the model. These methods require extensive training. PIC is the first approach to enhance long-form text generation capabilities from the perspective of position id and can be plugged into many LLMs without training.

3 Method

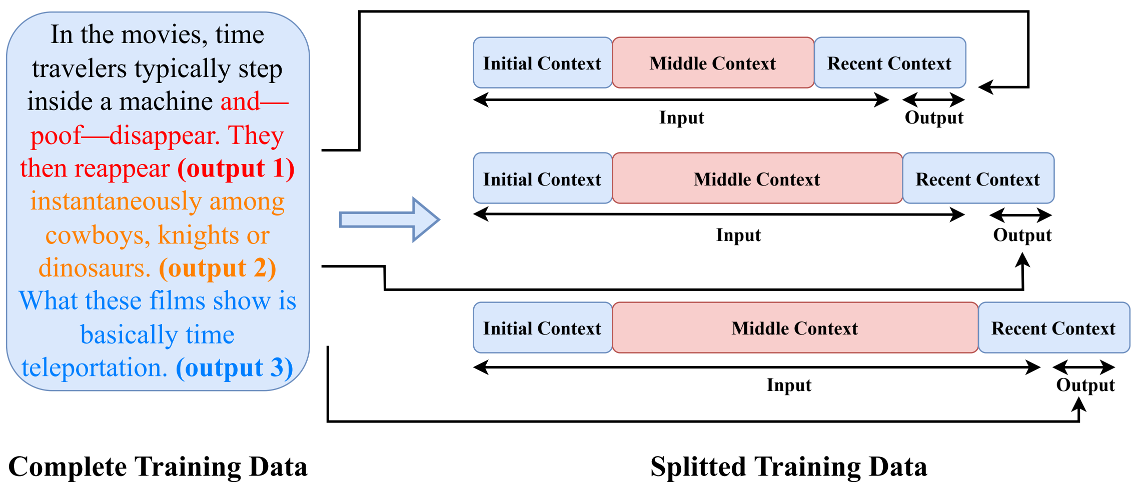
3.1 PIC

The core idea of PIC is straightforward: RoPE encodes relative positional information for tokens at different positions, if we compress the position ids of tokens so that the relative positions perceived by the models are smaller than the actual relative positions, the models may be able to generate longer content. In Section 4.2, experimental results demonstrate that this simple starting point achieves highly effective results. Based on this idea, we propose three variants of PIC: **Naive PIC**, **NTK-aware PIC**, and **Dynamic PIC**.

Naive PIC As shown in Figure 1(a), Naive PIC is defined as compressing the position ids of all positions equally by a fixed ratio. This is equivalent to modifying $g(m)$ in Equation (2) to:

$$g(m) = \frac{m}{s_{cr}}, \quad (7)$$

where m represents the actual position id of the current token and s_{cr} represents the **compression ratio**. The form of Naive PIC is identical to that of PI (Chen et al., 2023). However, the focus of PI is more on extending the context window, whereas the focus of Naive PIC is on enabling the model,



In the movies, time travelers typically step inside a machine and—poof—disappear. They then reappear (output 1) instantaneously among cowboys, knights or dinosaurs. (output 2) What these films show is basically time teleportation. (output 3)
Initial Context
Middle Context
Recent Context
Input
Output
Complete Training Data
Splitted Training Data

Figure 2: Illustration of Dynamic PIC-SFT. A complete text is split into segments, where the input for each subsequent segment is formed by concatenating the input and output of the previous segment.

which already has a sufficient context window, to generate more content.

NTK-aware PIC NTK-aware PIC indirectly compresses the position ids by modifying the RoPE Base b, thus altering $h(\theta_j)$ in Equation (2):

$$h(\theta_j) = (b \cdot s_{rr})^{-2(j-1)/d}, \tag{8}$$

where b is the original RoPE base, s_{rr} is the **RoPE base scaling ratio**, j denotes the positional index of the token vector, and d represents the total dimension size of the token vector. As shown in Figure 1(b), the position ids are not directly compressed. Instead, the RoPE base is doubled, which affects the values in different dimensions of the token vector.

Dynamic PIC As shown in Figure 1(c), an input context C consists of three parts: $C = \{\text{Initial Context, Middle Context, Recent Context}\}$. The core idea of Dynamic PIC is to compress the position ids of the middle context only while keeping the position ids of the recent context and the initial context unchanged. Assuming the current total context length is l_c, we have:

$$g(m) = \begin{cases} m & \text{if } m \le l_{ic} \text{ or } m \ge l_c - l_{rc}, \\ \frac{m}{s_{cr}} & \text{otherwise.} \end{cases} \tag{9}$$

Here, s_{cr} represents the **compression ratio**, l_{ic} represents the length of the initial context, and l_{rc} represents the length of the recent context.

In the practical implementation of Dynamic PIC, we need to modify the forward function of the model. Originally, the KV cache stores the K and V after applying RoPE. In Dynamic PIC, we only compress the position ids of the middle context, and for each next token prediction, we need to update the position ids of all preceding tokens. Therefore, in the Dynamic PIC implementation, we need to store the K and V before applying RoPE. As a result, the steps for applying RoPE and updating the KV cache need to be swapped, as shown in Algorithm 1.

Algorithm 1 Dynamic PIC

```
Require: Query $Q$, Key $K$, Value $V$, KV Cache $C$
Original Implementation
$Q, K \leftarrow$ apply_rope$(Q, K)$
if $C$ is not null then
    $K, V \leftarrow C$.update$(K, V)$
end if
Dynamic PIC Implementation
if $C$ is not null then
    $K, V \leftarrow C$.update$(K, V)$
end if
$Q, K \leftarrow$ apply_rope$(Q, K)$
```

3.2 PIC-SFT

To further enhance the long-form text generation capabilities of LLMs, we integrate PIC with SFT and propose **PIC-SFT**. **Naive PIC-SFT** is defined as compressing position ids to $1/s_{cr}$ of their original value during SFT, while **NTK-aware PIC-SFT** is defined as expanding the RoPE base to s_{rr} times its original value during SFT. In the cases of Naive PIC-SFT and NTK-aware PIC-SFT, all tokens are processed in the same way, allowing direct combination with SFT. However, for Dynamic PIC, the position ids of the context need to be dynamically updated for each new token prediction. If directly combined with SFT, this would significantly increase storage and computation overhead during training, making parallel computation infeasible. To address these challenges, we propose **Dynamic PIC-SFT**, an SFT approach tailored to Dynamic PIC.

As shown in Figure 2, we split a complete piece of text into segments of equal length, starting from the end of the text and moving backward. These segments serve as the output, while all preceding content is treated as the input. Since we only compute the loss for the output, this approach effectively computes the prediction loss for the entire text. This approach ensures that the position ids for each segment remain fixed, allowing each segment to be treated as an independent sample for parallel computation, significantly reducing computational overhead during training. Furthermore, since the splits are fine-grained, the fixed position ids for each segment closely align with the behavior of Dynamic PIC during inference, ensuring that the training stage closely approximates the actual inference stage.

4 Experiments

4.1 Experimental Setup

Models & Benchmarks In this work, we mainly evaluate 3 LLMs: LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Qwen-2.5-7B-Instruct (Yang et al., 2024), and GLM-4-9B-Chat (Zeng et al., 2022). To evaluate the performance of PIC in long-form text generation, we select HelloBench (Que et al., 2024) and LongBench-Write (Bai et al., 2024) as the evaluation benchmarks. As the training dataset, we have chosen the high-quality bilingual SFT dataset LongWriter-6k[1] which contains 6000 long-form instructional samples. We only selected samples from LongBench-Write with generation length requirements greater than 2000 as the test set. To ensure consistency with LongBench-Write, we focus on the "Heuristic Text Generation" subset of HelloBench for evaluation. Therefore, for PIC-SFT, HelloBench can be considered as out-of-distribution data, while LongBench-Write can be seen as in-distribution data, enabling a more comprehensive evaluation. To evaluate the model's performance on short-form text generation, we select MMLU (Hendrycks et al., 2020) and GSM8K (Cobbe et al., 2021). The number of test samples for HelloBench, LongBench-Write, MMLU, and GSM8K are 123, 47, 14042, and 1319, respectively.

Inference & Training To ensure reproducibility, all inference is conducted using greedy decoding. For training, we use LLaMA-Factory (Zheng et al., 2024) with a learning rate of 5e-5, warmup of 0.1, and 3 epochs. All experiments are conducted on 64 NVIDIA H100 80GB GPUs.

PIC In Section 4.3, for Dynamic PIC-SFT, the length of the initial context is 10, and the length of the recent context is 600. For NTK-aware PIC-SFT, s_{rr} is set to 16.

Metrics We focus on two aspects: generation length and generation quality. We adopt the metrics for length and quality as defined in HelloBench and LongBench-Write. Specifically, $\boldsymbol{S_l}$ represents the length score, $\boldsymbol{S_q}$ represents the quality score, and $\overline{\boldsymbol{S}}$ is their average. Additionally, we have observed that generating overly long texts often leads to severe repetition. We consider such repetitive generations as failed outputs, and thus, we calculate the **Corrupted Ratio** to evaluate the effectiveness of the generation. The corrupted ratio is defined as the number of corrupted samples divided by the total number of samples:

$$\text{CR} = \frac{n_{corrupted}}{n_{total}}, \tag{10}$$

where $n_{corrupted}$ and n_{total} represent the number of corrupted samples and total samples, respectively. For MMLU and GSM8K, the evaluation metric is accuracy. To save resources and ensure reproducibility of results, we use LLaMA-3.3-70B-Instruct[2] as the judge model, and the inference engine is vLLM (Kwon et al., 2023). More details are shown in Appendix A.

4.2 PIC

To verify the effectiveness and generalization of Naive PIC, NTK-aware PIC, and Dynamic PIC, we

[1]`https://huggingface.co/datasets/THUDM/LongWriter-6k`

[2]`https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct`

Table 1: The results of the three PIC methods, Naive PIC, NTK-aware PIC, and Dynamic PIC, with varying parameters are presented. Evaluation Benchmark is HelloBench. "CR" denotes Corrupted Ratio. The results for Quality and CR have been multiplied by 100. Only a subset of the results is shown here, more detailed experimental results can be found in Appendix B. The blue row represents the baseline.

Parameter	**LLaMA-3.1-8B-Instruct**				**Qwen-2.5-7B-Instruct**				**GLM-4-9B-Chat**			
	Length ↑	**Tokens ↑**	**Quality ↑**	**CR ↓**	**Length ↑**	**Tokens ↑**	**Quality ↑**	**CR ↓**	**Length ↑**	**Tokens ↑**	**Quality ↑**	**CR ↓**
Naive PIC, Parameter = Compression Ratio												
0.5	779.26	806.91	73.91	43.09	824.68	850.16	69.00	59.35	802.42	850.29	77.82	6.50
0.75	959.35	998.88	78.52	4.07	832.16	860.22	77.65	11.38	946.47	994.15	79.64	**0.81**
1	**1001.86**	**1055.42**	**79.85**	**1.63**	**958.33**	**990.54**	**80.17**	**4.07**	990.09	1039.23	**81.07**	2.44
2	691.76	725.72	71.96	13.01	858.95	905.18	71.51	54.47	**1120.43**	**1109.97**	71.17	46.34
NTK-aware PIC, Parameter = RoPE Base Scaling Ratio												
0.33	770.85	806.58	77.54	**0**	797.55	823.27	78.95	4.88	869.17	914.07	79.18	**0**
0.67	882.79	925.79	79.49	**0**	962.98	997.83	78.67	5.69	944.34	989.40	80.02	0.81
1	1020.80	1071.84	80.97	1.63	954.75	988.72	80.98	**4.07**	986.97	1031.88	81.03	2.44
4	1522.97	1604.71	**82.38**	7.32	1140.97	1178.38	80.98	6.50	1154.25	1203.4	82.16	1.63
16	**1802.11**	**1902.17**	81.05	34.15	**1391.56**	**1436.65**	**81.77**	13.01	**1352.04**	**1404.38**	**82.52**	1.63
Dynamic PIC, Parameter = Compression Ratio, Length of Initial Context = 4, Length of Recent Context = 200												
0.25	816.65	857.19	79.30	**0**	823.67	854.88	78.88	4.88	930.33	973.37	79.92	3.25
0.5	893.25	940.46	79.15	**0**	864.46	895.39	79.25	5.69	937.21	980.22	79.40	1.63
1	1024.05	1076.16	81.07	1.63	966.61	1001.67	80.19	**4.07**	984.50	1029.95	80.86	2.44
4	1380.63	1444.02	**81.31**	0.81	1125.13	1164.26	81.58	17.07	1154.62	1206.48	81.75	**0.81**
16	1938.24	2028.61	81.12	24.39	**1520.81**	**1569.01**	80.93	22.76	1347.92	1406.73	82.72	4.88
128	**1961.89**	**2101.06**	75.31	61.79	1490.53	1543.54	**81.96**	35.77	**1478.12**	**1538.12**	**83.43**	4.07

have tested the long-form text generation capabilities of the three models under different parameters. The experimental results are shown in Figure 1 and Table 1. Based on the experimental results, we draw the following conclusions:

(1). **Naive PIC performs poorly**: As observed in Table 5, when the compression ratio is not equal to 1, the generation quality tends to degrade significantly, with the quality score dropping to 0 and the corrupted ratio close to 100%. This result aligns with the conclusions from PI (Chen et al., 2023), where directly applying linear interpolation to position ids leads to undesirable outcomes.

(2). **NTK-aware PIC and Dynamic PIC perform well**: From the trend in Figure 1, it can be observed that as the Compression Ratio or RoPE Base Scaling Ratio increases, the generation length increases smoothly. At the same time, the generation quality does not deteriorate and even shows a slight improvement, which is consistent with our core idea: compressing position ids can stimulate the model's inherent ability for long-form text generation. This validates the effectiveness and generalization of PIC across three models.

(3). **Corrupted Ratio**: Based on Table 6 and Table 7, in most cases, the corrupted ratio remains below 20% (with s_{rr} ranging from 0.25 to 64 and s_{cr} ranging from 0.25 to 64). This indicates that within a certain range, NTK-aware PIC and Dynamic PIC do not negatively impact generation quality. This result confirms the superiority of NTK-aware PIC and Dynamic PIC compared to Naive PIC. However, there is a threshold for these ranges. When the threshold is exceeded (e.g., when s_{rr} exceeds 8 on LLaMA-3.1-8B-Instruct), the generation quality deteriorates significantly.

(4). **Generation Length**: The trend of generation length is entirely consistent with the generation token. Additionally, it can be observed that NTK-aware PIC and Dynamic PIC are able to extend the model's generated length to approximately 1.5 times the original length without compromising generation quality, and this is achieved without any additional training.

4.3 PIC-SFT

To further improve the long-form text generation capabilities of LLMs, we propose PIC-SFT and conduct experiments. The experimental results are shown in Table 2 and Figure 3. Based on the results, we summarize the following conclusions:

(1). **SFT improves performance significantly**: Fine-tuning with high-quality instruction datasets can significantly enhance the long-form text generation capabilities of the model. Regardless of the specific SFT method, fine-tuning improves both the generation length and generation quality. On HelloBench, the generation length increases by 4–5 times, and the generation quality improves from 82.25 to 89.26 (on LLaMA-3.1-8B-Instruct).

Table 2: The experimental results of different SFT methods. "CR" denotes Corrupted Ratio. All experimental results maintain consistency between training and inference, for example, the inference method for NTK-aware PIC-SFT is NTK-aware PIC.

Method	HelloBench-HTG			LongBench-Write			
	Length ↑	Quality ↑	CR ↓	S_l ↑	S_q ↑	$\overline{S}$ ↑	CR ↓
LLaMA-3.1-8B-Instruct							
w/o SFT	1040.89	82.25	**1.63**	18.87	90.33	54.60	3.33
SFT	4342.24	86.87	3.25	78.06	**98.73**	88.40	**1.67**
NTK-aware PIC-SFT	4737.44	89.17	4.07	77.71	97.80	87.75	**1.67**
Dynamic PIC-SFT	**5032.98**	**89.26**	4.07	**82.08**	97.00	**89.54**	**1.67**
Qwen-2.5-7B-Instruct							
w/o SFT	965.51	80.58	4.07	41.59	98.12	69.85	8.33
SFT	4070.64	87.85	3.25	81.92	**99.64**	90.78	3.33
NTK-aware PIC-SFT	3964.59	87.46	8.13	81.13	98.48	89.81	1.67
Dynamic PIC-SFT	**4282.14**	**89.39**	**0.81**	**84.15**	99.57	**91.86**	**0.83**
GLM-4-9B-Chat							
w/o SFT	967.32	81.09	**2.44**	27.03	95.65	61.34	3.33
SFT	4057.83	**84.05**	14.63	**77.86**	96.94	87.40	8.33
NTK-aware PIC-SFT	3910.37	81.82	30.08	77.52	94.17	85.84	20.0
Dynamic PIC-SFT	**4357.59**	82.57	3.25	76.48	**98.7**	**87.59**	**0**

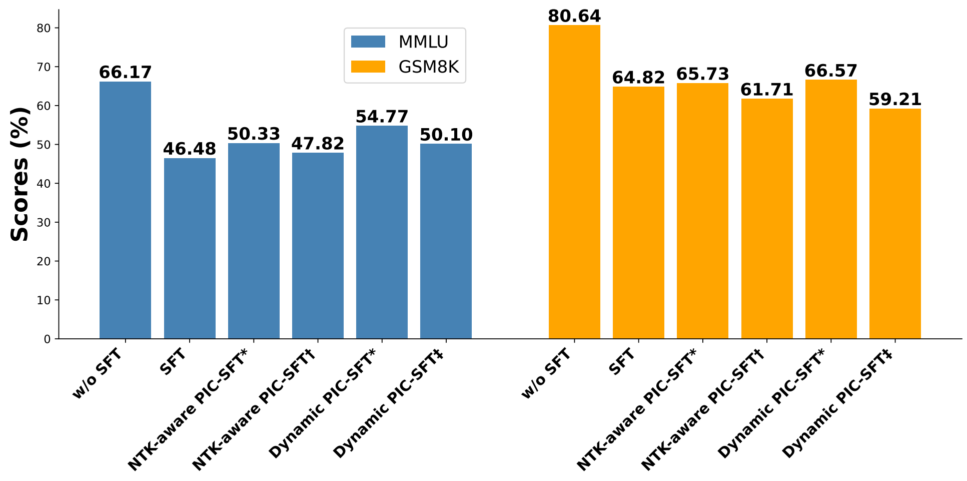


Figure 3: Results of different SFT methods evaluated on MMLU and GSM8K. The base model is LLaMA-3.1-8B-Instruct. ∗ represents standard inference, † represents NTK-aware PIC inference, and ‡ represents Dynamic PIC inference.

(2). **Dynamic PIC-SFT performs best**: As shown in Table 2, Dynamic PIC-SFT achieves the best results in both generation length and generation quality, whether for in-distribution benchmark LongBench-Write or out-of-distribution benchmark HelloBench. Furthermore, Dynamic PIC-SFT does not compromise the validity of the generated content. For example, for GLM-4-9B-Chat, both SFT and NTK-aware PIC-SFT result in reduced validity (the corrupted ratio on HelloBench is 14.63 and 30.08, and on LongBench-Write is 8.33 and 20.0, respectively). In contrast, the corrupted ratio for Dynamic PIC-SFT is only 3.25 on HelloBench and 0 on LongBench-Write.

(3). **The decoupling of PIC-SFT**: Figure 3 shows the performance of different methods on MMLU and GSM8K. Since the fine-tuning focuses solely on long-form text generation tasks, the overall output distribution tends to favor longer content. As a result, the model's ability for shorter text generation may decline, leading to lower scores on tasks such as MMLU and GSM8K. However, an interesting observation arises when the PIC-SFT model is used for standard (i.e., non-PIC) infer-

Table 3: Ablation Study on Dynamic PIC. The base model is LLaMA-3.1-8B-Instruct. “IC” denotes initial context, “RC” denotes recent context. The blue row represents the baseline.

Parameter	**HelloBench-HTG**			**LongBench-Write**			
	Length ↑	**Quality** ↑	**CR** ↓	S_l ↑	S_q↑	$\overline{S}$ ↑	**CR** ↓
Ablation study on the modules of Dynamic PIC							
Baseline	1401.97	81.13	0.81	27.73	4.36	16.05	5.83
w/o IC	1400.53	81.45	1.63	29.32	4.44	16.88	6.67
w/o RC	306.11	15.3	78.05	2.14	1.39	1.76	48.33
w/o IC&RC	266.61	10.72	77.24	0	1.43	0.72	45.0

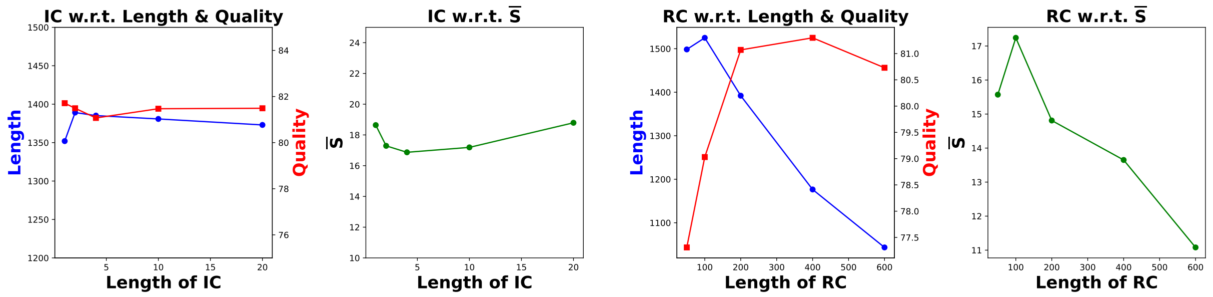


Figure 4: Ablation Study on length of IC and length of RC.

ence. In this case, it achieves higher scores on MMLU and GSM8K. In addition, we have conducted additional experiments to demonstrate this, as detailed in Appendix C. These Phenomena indicate that PIC-SFT exhibits decoupling. Specifically, in NTK-aware PIC, the RoPE Base value is modified, while in Dynamic PIC, the position ids of the middle context are compressed. When training with NTK-aware PIC-SFT, standard inference retains more of the model’s original capabilities, while NTK-aware PIC Inference focuses more on long-form text generation tasks. This observation suggests the potential for decoupling a model’s abilities in short-form text generation and long-form text generation through PIC-SFT.

4.4 Ablation Study on Dynamic PIC

To investigate the effectiveness of each module in Dynamic PIC, we have conducted an ablation study, with the results shown in Table 3 and Figure 4. It can be observed that the Recent Context (**RC**) is crucial for Dynamic PIC. Removing RC leads to a significant drop in both generation length and generation quality, as well as an increase in the corrupted ratio, resulting in an overall decline in output performance. As for the Initial Context (**IC**), its impact on the long-form text generation tasks is not significant. However, based on the conclusions from StreamingLLM (Xiao et al., 2023), the Initial Context is critical for long-context understanding. For consistency, we choose to include the Initial Context by default. Lastly, the length of RC has a noticeable effect on the generation length and generation quality. As shown in Figure 4, when the length of RC is smaller, the model tends to produce longer outputs, but the generation quality decreases.

5 Conclusion

In this work, we introduce Position ID Compression (PIC), a simple, efficient, and plug-in approach to enhance the long-form text generation capabilities of LLMs. By compressing position ids, PIC enables models to generate longer content without additional training. We propose two improved variants, NTK-aware PIC and Dynamic PIC, which achieve better efficiency and flexibility. To further improve long-form text generation, we combine PIC with SFT and propose PIC-SFT, which achieves up to a 5x increase in generation length while maintaining the generation quality. Through extensive experiments, we show that PIC-SFT achieves state-of-the-art performance on both in-distribution and out-of-distribution datasets, with significantly reduced corrupted ratios compared to other methods. Additionally, the observed ability of PIC-SFT to decouple short-text and long-text capabilities highlights its potential for more flexible deployment in future LLM applications. These findings suggest that proper design and configuration of position id compression strategies are key to unlocking the full potential of long-form text generation in LLMs.

Limitations

The choice of the judge model The quality evaluation for HelloBench and LongBench-Write involves the use of the judge model. In this work, to ensure the reproducibility of results while saving resources, we selected the latest open-source model, LLaMA-3.3-70B-Instruct, as our judge model. However, in the original works of HelloBench and LongBench-Write, GPT-4o was used for evaluation. In the future, we plan to conduct evaluations using GPT-4o and provide the corresponding results.

Further exploration on the decoupling of PIC-SFT In Section 4.3, we have discussed the decoupling phenomenon of PIC-SFT. This interesting finding suggests that the model's capabilities might be decoupled and expressed through different inference methods. However, in this work, we focus on identifying and explaining this phenomenon, without conducting further exploration. We consider further investigation of this phenomenon as future work.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62477001.

References

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

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023a. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023b. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longwriter: Unleashing 10,000+ word generation from long context llms. *arXiv preprint arXiv:2408.07055*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

bloc97. 2023a. Add ntk-aware interpolation "by parts" correction.

bloc97. 2023b. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2024. Security and privacy challenges of large language models: A survey. *arXiv preprint arXiv:2402.00888*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

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

Md Meftahul Ferdaus, Mahdi Abdelguerfi, Elias Ioup, Kendall N Niles, Ken Pathak, and Steven Sloan. 2024. Towards trustworthy ai: A review of ethical and robust large language models. *arXiv preprint arXiv:2407.13934*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.

Jian Guan, Zhuoer Feng, Yamei Chen, Ruilin He, Xiaoxi Mao, Changjie Fan, and Minlie Huang. 2022. Lot: A story-centric benchmark for evaluating chinese long text understanding and generation. *Transactions of the Association for Computational Linguistics*, 10:434–451.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Pedram Hosseini, Jessica M Sin, Bing Ren, Bryceton G Thomas, Elnaz Nouri, Ali Farahanchi, and Saeed Hassanpour. 2024. A benchmark for long-form medical question answering. *arXiv preprint arXiv:2411.09834*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9.

Bin Lin, Chen Zhang, Tao Peng, Hanyu Zhao, Wencong Xiao, Minmin Sun, Anmin Liu, Zhipeng Zhang, Lanbo Li, Xiafei Qiu, et al. 2024. Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache. *arXiv preprint arXiv:2401.02669*.

Xiang Liu, Peijie Dong, Xuming Hu, and Xiaowen Chu. 2024a. Longgenbench: Long-context generation benchmark. *arXiv preprint arXiv:2410.04199*.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, et al. 2024b. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 38–56.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following for long-form text generation. *arXiv preprint arXiv:2406.19371*.

Shanghaoran Quan, Tianyi Tang, Bowen Yu, An Yang, Dayiheng Liu, Bofei Gao, Jianhong Tu, Yichang Zhang, Jingren Zhou, and Junyang Lin. 2024. Language models can self-lengthen to generate long texts. *arXiv preprint arXiv:2410.23933*.

Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, et al. 2024. Hellobench: Evaluating long text generation capabilities of large language models. *arXiv preprint arXiv:2409.16191*.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539.

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

Haochen Tan, Zhijiang Guo, Zhan Shi, Lu Xu, Zhili Liu, Yunlong Feng, Xiaoguang Li, Yasheng Wang, Lifeng Shang, Qun Liu, et al. 2024. Proxyqa: An alternative framework for evaluating long-form text generation with large language models. *arXiv preprint arXiv:2401.15042*.

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

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al.

2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, et al. 2024. Long-form factuality in large language models. *arXiv preprint arXiv:2403.18802*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. Longproc: Benchmarking long-context language models on long procedural generation. *arXiv preprint arXiv:2501.05414*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*.

A Details of Metrics

For HelloBench and LongBench-Write, we use different metrics to represent the generation length and generation quality. For HelloBench, we measure generation quality using the HelloEval scores from HelloBench, which are referred to as "Quality" in the experimental table for simplicity. It is worth noting that we do not adopt Score Rescaling as described in the original paper. For LongBench-Write, we use the $\overline{\mathrm{S}}$ metric as defined by Bai et al. (Bai et al., 2024) to represent the overall generation quality. For MMLU evaluation, we use 5-shot, and for GSM8K evaluation, we use 4-shot combined with Chain-of-Thought (CoT).

In addition, we have used the Corrupted Ratio to measure the model's effective generation rate. Since different models may have varying numbers of corrupted samples, we ensure comparability by following a specific approach. For models with a Corrupted Ratio of less than 20%, we have calculated the average of non-corrupted examples shared among compared models as the final experimental result, ensuring direct comparability. For models with a Corrupted Ratio greater than 20%, including their corrupted examples in the shared non-corrupted set would result in too few examples for averaging, diminishing comparability. Therefore, for models with a Corrupted Ratio above 20%, we have computed their averaged results independently. We use the NLTK (Bird, 2006) library to calculate the generated length. Additionally, we have observed that most repetitive samples do not end with the typical end-of-token of LLMs. Therefore, we determine whether a sample is corrupted based on regular expressions and whether it ends with the end-of-token.

Through comparisons, we have observed that the error introduced by considering or not considering the shared non-corrupted examples is less than 5%. However, for the sake of rigor, we provide this clarification.

B Additional Experimental Results

Table 4: Results of different SFT methods evaluated on MMLU and GSM8K. The base model is LLaMA-3.1-8B-Instruct. $*$ represents standard inference, † represents NTK-aware PIC inference, and ‡ represents Dynamic PIC inference.

Method	**MMLU**	**GSM-8K**
w/o SFT	66.17	80.64
SFT	46.48	64.82
NTK-aware PIC-SFT*	50.33	65.73
NTK-aware PIC-SFT†	47.82	61.71
Dynamic PIC-SFT*	54.77	66.57
Dynamic PIC-SFT‡	50.10	59.21

Table 5: The experimental results of Naive PIC with varying Compression Ratio are presented. "CR" denotes Corrupted Ratio. The blue row represents the baseline.

Compression Ratio	LLaMA-3.1-8B-Instruct				Qwen-2.5-7B-Instruct				GLM-4-9B-Chat			
	Length	Tokens	Quality	CR	Length	Tokens	Quality	CR	Length	Tokens	Quality	CR
0.25	0	0	0	100	0	0	0	100	0	0	0	100
0.33	0	0	0	100	0	0	0	100	0	0	0	100
0.5	779.26	806.91	73.91	43.09	824.68	850.16	69	59.35	802.42	850.29	77.82	6.5
0.67	904.13	947	77.91	6.5	841.76	869.94	77.8	13.01	896.15	945.18	78.87	1.63
0.75	959.35	998.88	78.52	4.07	832.16	860.22	77.65	11.38	946.47	994.15	79.64	**0.81**
1	**1001.86**	**1055.42**	**79.85**	**1.63**	958.33	990.54	**80.17**	**4.07**	990.09	1039.23	**81.07**	2.44
2	691.76	725.72	71.96	13.01	858.95	905.18	71.51	54.47	1120.43	1109.97	71.17	46.34
3	593.78	618.3	43.9	43.9	876.75	917.61	66.44	77.24	1233.74	1288.02	60.05	65.85
4	266.61	279.64	10.72	77.24	**1660**	**1721.57**	54.1	94.31	2304.38	2453.56	24.3	84.55
8	27.67	28.76	0	22.76	0	0	0	100	**5937.6**	**6642.4**	0	95.12
16	6.63	7.88	0	2.44	72.27	85.45	0	91.06	1923.5	2076	0	98.37

Table 6: The experimental results of NTK-aware PIC with varying RoPE Base Scaling Ratio are presented. "RBSR" denotes RoPE Base Scaling Ratio. "CR" denotes Corrupted Ratio. The blue row represents the baseline.

RBSR	LLaMA-3.1-8B-Instruct				Qwen-2.5-7B-Instruct				GLM-4-9B-Chat			
	Length	Tokens	Quality	CR	Length	Tokens	Quality	CR	Length	Tokens	Quality	CR
0.25	693.05	721.01	76.45	0.81	812.38	837.8	77.14	4.07	866.12	909.25	79.39	0.81
0.33	770.85	806.58	77.54	**0**	797.55	823.27	78.95	4.88	869.17	914.07	79.18	**0**
0.5	849.8	888.56	78.97	0.81	890.93	922.84	78.91	6.5	909.26	954.31	79.62	0.81
0.67	882.79	925.79	79.49	**0**	962.98	997.83	78.67	5.69	944.34	989.4	80.02	0.81
0.75	924.98	972.79	80.79	0.81	866.75	897.66	78.89	5.69	980.88	1025.65	80.86	0.81
1	1020.8	1071.84	80.97	1.63	954.75	988.72	80.98	4.07	986.97	1031.88	81.03	2.44
2	1259.69	1326.76	81.59	**0**	1042.36	1080.24	80.22	**3.25**	1051.68	1096.56	81.62	2.44
3	1381.08	1454.8	82.34	6.5	1092.58	1128.23	80.29	8.13	1106.7	1156.73	81.52	**0**
4	1522.97	1604.71	**82.38**	7.32	1140.97	1178.38	80.98	6.5	1154.25	1203.4	82.16	1.63
8	1675.86	1765.35	81.67	22.76	1256.05	1303.24	81.13	9.76	1245.97	1294.94	82.88	0.81
16	1802.11	1902.17	81.05	34.15	1391.56	1436.65	**81.77**	13.01	1352.04	1404.38	82.52	1.63
32	**1811.1**	**1917.76**	80.77	41.46	1519.52	1573.89	81.27	22.76	1453.24	1505.29	83.98	2.44
64	1676.64	1812.77	77.61	61.79	1585.94	1651.06	80.71	47.15	1605.64	1666.28	**84.03**	4.07
128	1493.38	1583.02	77.86	61.79	1761.41	1832.64	80.11	46.34	1701.72	1765.42	83.98	9.76
256	1312.87	1382.3	73.13	81.3	1776.43	1867.52	77.41	62.6	1798.69	1865.31	83.05	23.58
512	1066.64	1118.73	73.45	91.06	**1900.31**	**2006.42**	74.84	70.73	**1880.8**	**1953.73**	81.86	31.71

Table 7: The experimental results of Dynamic PIC with varying Compression Ratio are presented. "CR" denotes Corrupted Ratio. The blue row represents the baseline.

Compression Ratio	LLaMA-3.1-8B-Instruct				Qwen-2.5-7B-Instruct				GLM-4-9B-Chat			
	Length	Tokens	Quality	CR	Length	Tokens	Quality	CR	Length	Tokens	Quality	CR
0.25	816.65	857.19	79.3	**0**	823.67	854.88	78.88	4.88	930.33	973.37	79.92	3.25
0.33	858.72	902.11	79.84	1.63	858.68	889.78	78.36	**4.07**	899.69	941.86	80.5	**0.81**
0.5	893.25	940.46	79.15	**0**	864.46	895.39	79.25	5.69	937.21	980.22	79.4	1.63
0.67	945.2	994.1	80.16	1.63	883.42	916.3	78.91	5.69	931.02	974.52	81.11	1.63
0.75	982.13	1033.77	80.23	0.81	902	933.7	79.13	5.69	951.4	993.65	80.63	**0.81**
1	1024.05	1076.16	81.07	1.63	966.61	1001.67	80.19	**4.07**	984.5	1029.95	80.86	2.44
2	1155.42	1211.42	**81.89**	1.63	1075.7	1112.8	80.12	8.13	1051.12	1101.73	81.11	1.63
3	1283.81	1346.92	81.54	0.81	1104.83	1144.6	79.55	13.01	1130.18	1179.9	82	**0.81**
4	1380.63	1444.02	81.31	0.81	1125.13	1164.26	81.58	17.07	1154.62	1206.48	81.75	**0.81**
8	1656.45	1734.45	81.88	6.5	1282.11	1324.32	**82.35**	13.82	1234.58	1288.33	82.27	5.69
16	1938.24	2028.61	81.12	24.39	**1520.81**	**1569.01**	80.93	22.76	1347.92	1406.73	82.72	4.88
32	1919.77	2022.68	76.68	47.15	1479.34	1530.09	81.58	30.08	1428.65	1487.96	82.88	5.69
64	1817.23	1916.1	76.08	60.98	1454.23	1501.58	79.96	31.71	1441.52	1502.96	**83.45**	4.88
128	**1961.89**	**2101.06**	75.31	61.79	1490.53	1543.54	81.96	35.77	1478.12	1538.12	83.43	4.07
256	1737.69	1858.37	74.45	71.54	1471.03	1526.04	79.8	38.21	1462.22	1523.08	82.91	4.07
512	1576.7	1691.36	75.23	73.17	1460.7	1510.07	80.89	38.21	**1507.17**	**1572.32**	83.4	4.07

C The impact of different inference methods

We further conduct experiments on the different inference methods on PIC-SFT, the experimental results are shown in Table 8. We observe that, compared to NTK-aware PIC Inference, standard inference tends to generate shorter content with lower generation quality. Similarly, compared to Dynamic PIC Inference, standard inference also produces shorter content and lower generation quality. This indirectly indicates that after NTK-aware PIC-SFT, the long-form text generation capability of standard inference weakens. This shows that PIC Inference remains effective after SFT, consistent with the trend identified in Section 4.2.

D Potential Risks

LLMs have been observed to exhibit inherent biases, generating content that may contain discrimination in various aspects such as politics, gender, and race (Das et al., 2024; Ferdaus et al., 2024) due to biased training data. The harmful stereotypes manifested in the generated content can contribute to the oppression of those at social margins (Weidinger et al., 2021). Therefore, in various long-form text generation fields such as creative writing and story continuation, it is crucial to ensure that the relevant long texts generated by LLMs do not contain harmful stereotypes. Additionally, LLMs are prone to hallucinations, often generating information that is factually incorrect or non-existent (Huang et al., 2023; Sahoo et al., 2024). This issue is particularly prominent in applications requiring high accuracy, such as academic paper editing and news writing, where the dissemination of incorrect information can have serious consequences. Ensuring that LLMs generate reliable and accurate long texts is essential to maintain the credibility of the generated content.

E Computational Budget for PIC-SFT

The computation budget for NTK-aware PIC-SFT is 20 GPU hours and the computation budget for Dynamic PIC is 60 GPU hours.

Table 8: The experimental results of different SFT methods with different inference methods. The base model is LLaMA-3.1-8B-Instruct. $*$ represents standard inference, † represents NTK-aware PIC inference, and ‡ represents Dynamic PIC inference.

Method	**HelloBench-HTG**			**LongBench-Write**			
	Length $\uparrow$	**Quality** $\uparrow$	**CR** $\downarrow$	$S_l \uparrow$	$S_q \uparrow$	$\overline{S} \uparrow$	**CR** $\downarrow$
SFT*	4207.51	87.37	3.25	76.8	98.37	87.59	1.67
SFT†	5166.1	88.58	21.14	74.03	98.13	86.08	13.33
SFT‡	4632.37	87.95	3.25	80.02	97.97	89.0	0.0
NTK-aware PIC-SFT*	3717.72	87.34	12.2	80.29	97.32	88.80	5.83
NTK-aware PIC-SFT†	4582.63	89.03	4.07	80.08	97.56	88.82	1.67
Dynamic PIC-SFT*	4413.57	87.01	3.25	75.34	97.56	86.45	3.33
Dynamic PIC-SFT‡	5098.45	89.48	4.07	82.49	96.99	89.74	1.67