APE: Faster and Longer Context-Augmented Generation via Adaptive Parallel Encoding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many adaptive language model applications, such as RAG and ICL, require the efficient combination of multiple external contexts to generate a response. In this work, we explore the potential of parallel encoding to speedup generation and extend context by pre-caching the KV states of each context separately for direct loading and position reuse during inference. However, directly applying it reduces performance due to its misalignment with sequential encoding. To address this challenge, we propose APE, which brings a shared prefix, additional scaling factor, and lower attention temperature to align these two distributions of attention weights. Extensive experiments showcase APE improves performance by 7.8% over standard parallel encoding and 2.9% over sequential encoding for long contexts, while maintaining 93% accuracy in few-shot learning. For the efficiency evaluation, APE achieves a 976$\times$ speedup for 512K context-augmented generation with a 256-token response.

1 Introduction

Recently, retrieval-augmented generation (RAG) [11, 9] and in-context learning (ICL) [7, 23] have been widely adopted in large language models (LLMs) [8, 1] to drastically increase their ability to generalize to unseen tasks with external data. Both techniques incorporate a *sequential encoding* process to ground LLM inputs with additional texts: concatenating them into one sequence, and encoding the sequence into key-value (KV) states to serve as the context for the following query. This new, significantly longer input prompt enhances performance but introduces two key challenges as illustrated in Figure 1. First, the increased latency required to prefill these contexts becomes a bottleneck in many tasks [3, 2, 13]. Second, the limited size of context window [4] leads to an accuracy gap compared to the ideal case where all relevant texts are included.

One natural direction to address these issues is to enable *parallel encoding* [18, 26, 16, 19] of independent contexts. Specifically, each context can be encoded separately and the query attends to the KV states from all contexts during generation. This approach brings two benefit: (i) We can pre-cache KV states from all contexts for faster inference. (ii) We can reuse the positions across contexts, enable more contexts to fit into the limited context window.

Despite these potential advantages, Figure 2 demonstrate that directly applying parallel encoding reduces performance in RAG and ICL, with average declines of 4.9% (with longer context) and 49.0%, respectively. To address this challenge, we propose Adaptive Parallel Encoding (**APE**), a simple yet effective method to enable efficient and accurate parallel context encoding for LLM inference. First, we observe an inherent alignment between sequential and parallel encoding. Our key insight, as shown in Figure 3 and 4, is that *KV states from independently encoded contexts can be naturally merged into a sequence due to their similarity in both direction and magnitude, attributed to the presence of an attention sink* [24]. Next, we identify the remaining misalignments in attention weights and re-align them to sequential encoding (Figure 1 (bottom)). The main contribution of this paper is as follows:

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

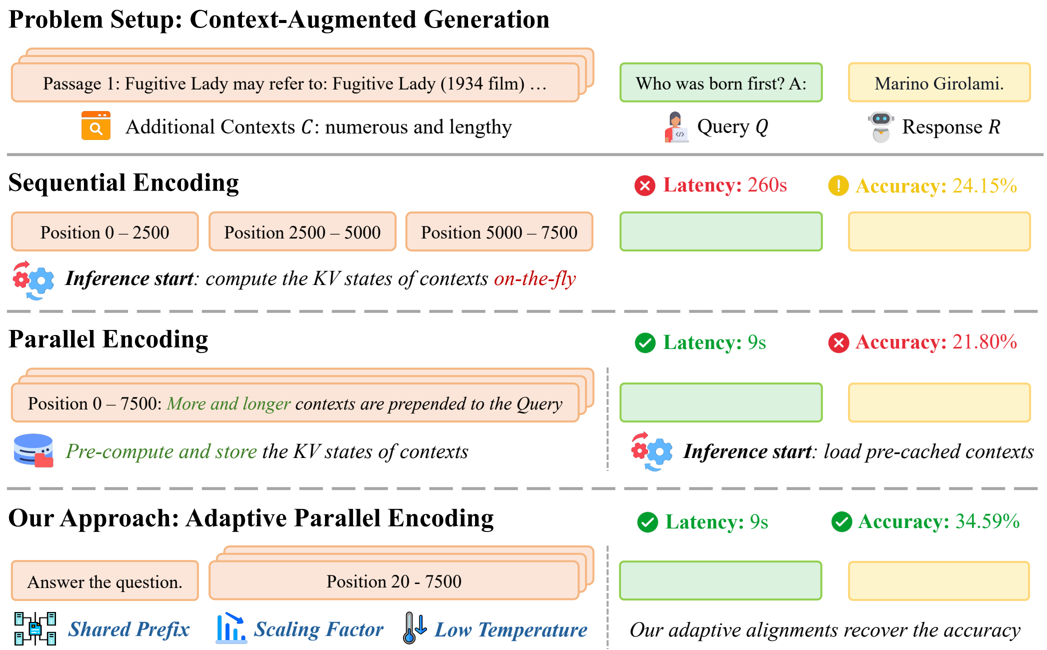
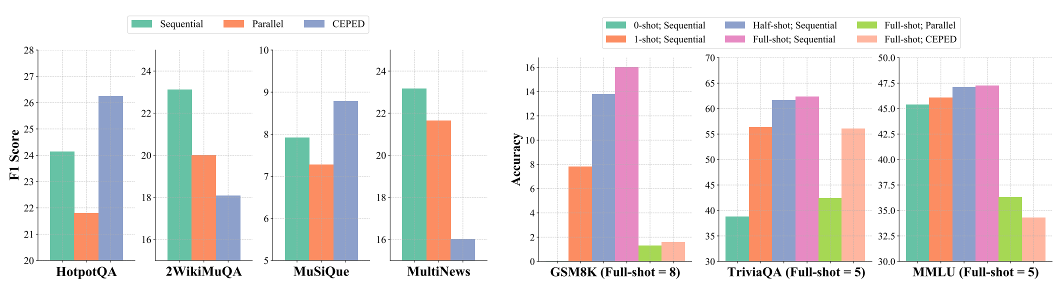


Figure 1: **Overview of our approach.** Context-augmented generation leverages additional context for response generation, but faces challenges with sequential encoding (exceeding LLM's window size and increasing prefilling time) and parallel encoding (worse performance). We propose adaptive parallel encoding, which aligns attention weight distribution with sequential encoding through shared prefix, scaling factor, and adaptive temperature. Our approach brings the benefit of parallel encoding while preserving the accuracy of the prediction without requiring additional fine-tuning of the model.

- We systematically analyze the distribution properties of parallel encoding, focusing on KV states across samples and positions and identifying alignments and misalignments with sequential encoding.
- We introduce APE with three key alignment steps: (i) Prepend a shared prefix to each context, avoiding the duplication of abnormal token distributions that can occur at the start positions. (ii) Apply a scaling factor to offset the increased attention weights resulting from placing all contexts closer to the query. (iii) Utilize a lower attention temperature to focus on undervalued semantically important tokens.
- We empirically show that APE (i) outperforms parallel encoding by 7.8%; (ii) extends the context length and surpasses sequential encoding by 2.9% in long-context scenarios; (iii) maintains 93% accuracy of sequential encoding using same input; (iv) accelerates long-context generation up to 976×.

2 Observations



(a) Long-context Understanding

(b) Few-shot Learning

Figure 2: Performance comparison of sequential encoding, parallel encoding, and CEPED in RAG and ICL.

In this section, we evaluate sequential encoding, parallel encoding, and CEPE-Distilled (CEPED) [26] on various tasks using LLAMA-2-7B-CHAT [1], with results in Figure 2. First, trainable approaches like CEPED fail on most tasks, indicating poor generalization to complex problems. Second, despite LLMs being trained sequentially, parallel encoding does not break down. We explore this phenomenon to elucidate both the alignments and misalignments between parallel and sequential encoding.

[1]We use LLAMA-2 for CEPED, as it's the only supported model. For other analyses, we employ LLAMA-3.

2.1 Comparing Parallel Encoding and Sequantial Encoding.

Our analysis reveals that the attention mechanism in LLMs naturally enables the comparison and combination of KV states from different contexts. To clarify this, Figures 3 and 4 visualize the direction and magnitude of KV states across different samples and positions, where we observe that key states share similar directions and value states demonstrate comparable magnitudes across different contexts.

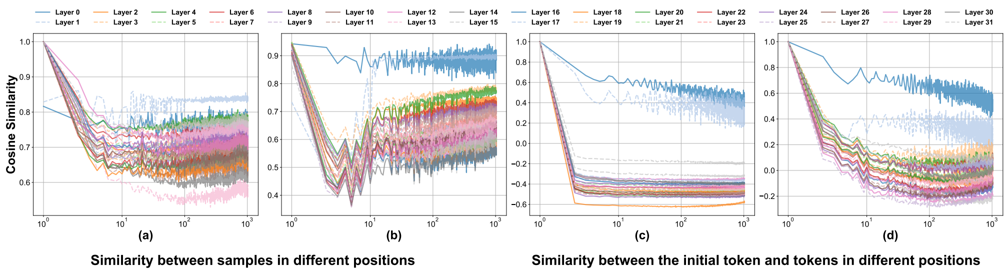


Figure 3: **Left:** Both LLAMA-3-8B-INSTRUCT (a) and MISTRAL-7B-INSTRUCT-V0.3 (b) exhibit a cosine similarity larger than 0.9 between the key states from distinct initial tokens. **Right:** Initial token's key states show similar negative similarity to those from other positions for each model. The X-axis is the position in the context, using a logarithmic scale. Results are measured on HotPotQA.

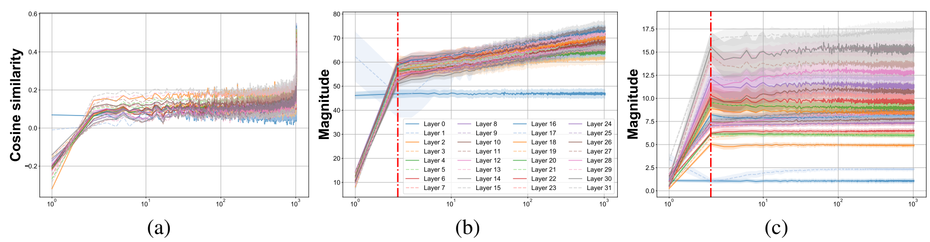


Figure 4: **Left:** The cosine similarity between query and key states increases as the distance between their positions decreases. **Middle and Right:** The magnitude of key and value states remain relatively stable across positions, with the exception of an abnormal region demarcated by a red dashed line.

Key states from different contexts are similar. In Figures 3a and 3b, we measure the cosine similarity of the key states between different initial tokens for various models, which consistently yields a value close to 1. This indicates that the direction of the initial token remains largely invariant across different input. Figure 3c and 3d further visualize the similarity between initial token's key states and those from subsequent positions, where we observe comparable negative values, showing a similarly large angle between initial key states and following ones. These findings, combined with the small variance in Figure 4, showcase that key states from different contexts share similar directions and magnitudes.

Values states from different contexts are similar. In the $\mathrm{Softmax}$ attention, value states are combined through a weighted summation. This normalization determines the magnitude of current value states based on those from all positions across contexts, yielding a similar L^2 norm, as shown in Figure 4c.

Opportunities for improvement. Previous analyses show that KV states exhibit a natural alignment across contexts. However, the residual misalignments still severely reduce performance:

- In Figures 3 and 4, we observe a notable discrepancy in both direction and magnitude within the first few positions. We designate this area as an abnormal region within the whole context.
- Figure 4a illustrates the cosine similarity between query states from the last position and all key states. A significant increase is observed when the distance between these states is extremely close.

3 Adaptive Parallel Encoding

With all the lessons learned in Section 2, we will design our Adaptive Parallel Encoding to address the misalignments, enabling a training-free shift to parallel encoding with minimal performance drop.

3.1 Prepending Shared Prefix.

Figure 4b and 4c show that the magnitudes of KV states for the initial tokens differ significantly from those of subsequent tokens. This discrepancy poses a challenge when encoding contexts in parallel from the beginning. To address this, we prepend a shared prefix to all contexts, ensuring these KV states appear only once per generation step. The choice of prefix depends on the model and task; we use existing system prompts and instructions when available, or insert newline characters before contexts otherwise.

3.2 Adding Scaling Factor.

In Figure 4a, the cosine similarity between query and key states increases as their distance decreases, with a notably sharper rise when the distance approaches zero. To offset this, we introduce a scaling factor s smaller than one to the context. This factor is applied after the $\exp$ operation in the $\mathrm{Softmax}$, allowing for a proportionally greater reduction for larger product values between query and key states.

3.3 Adjusting Attention Temperature.

We adjust the attention temperature T to a value less than 1 to emphasize semantically important tokens whose attention weights are above average but not as high as those closest to the query token. A carefully chosen temperature can recover the attention on these tokens while still maintaining a reduced scaling for the query's immediate neighbors. To prevent an overall increase in attention weights across the entire context, we also apply the temperature T as an exponent to the sum of attention weights before normalization. This can be expressed as $(\sum \exp(qk/T))^T$, where q and k ar query and key states, respectively.

3.4 Formulation.

Given these steps, we formulate the attention in APE from the standard $\mathrm{Softmax}$ attention (ignore $\sqrt{d}$), where Q, K, and V are the query, key, and value states from the input, and C_i denotes i-th context.

$$O = \mathrm{Softmax}(Q[K_{C_0}^\top,...,K_{C_{N-1}}^\top,K^\top]) \times [V_{C_0},...,V_{C_{N-1}},V] \tag{1}$$

$$= \frac{[A_{C_0},...,A_{C_{N-1}},A]}{\sum_{i=0}^{N-1}\sum_{j=0}^{l_i-1} a_{C_i,j} + \sum_{j=0}^{l-1} a_j} \times [V_{C_0},...,V_{C_{N-1}},V], \tag{2}$$

where $A_{C_i} = [\exp Qk_{C_i,0}^\top,...,\exp Qk_{C_i,l_i-1}^\top]$ and $a_{C_i,j} = \exp Qk_{C_i,j}^\top$. Similar for A and a_j.

After incorporating our proposed changes, the formula for our refined attention calculation becomes:

$$O' = \frac{[A_P,A'_{C_0},...,A'_{C_{N-1}},A]}{\sum_{j=0}^{l_P-1} a_{P,j} + (\sum_{i=0}^{N-1}\sum_{j=0}^{l_i-1} a'_{C_i,j})^T + \sum_{j=0}^{l-1} a_j} \times [V_P,V_{C_0},...,V_{C_{N-1}},V], \tag{3}$$

where $A'_{C_i} = [s \cdot \exp Qk_{C_i,0}^\top/T,...,s \cdot \exp Qk_{C_i,l_i-1}^\top/T]$ and $a'_{C_i,j} = s \cdot \exp Qk_{C_i,j}^\top/T$.

Here, A_P represent the attention weights for the shared prefix, respectively. The scaling factor s and attention temperature T for the context are both less than 1 ($s < 1, T < 1$). All these modifications can be fused into fast attention implementations such as [6] without additional cost.

4 Experiments

4.1 Long-context Understanding

Setup. Our evaluation involves four tasks with multi-document input on LongBench [3] and three models limited in context length: LLAMA-3-8B-INSTRUCT [8], LLAMA-2-7B-CHAT [22], and GEMMA-2-9B-IT [20]. Baselines include: (i) *Prompt Compression*: LLMLingua2 [17], (ii) *KV Cache Eviction*: StreamingLLM [24], (iii) *Long-context FT* [10, 21], (iv) *Parallel Encoding*: PCW [18], CEPE [26].

Results. As in Table 1, APE is the only method that consistently enhances performance across various models, leading to an average improvement of 2.9% compared to the base models. Moreover, it can generalize to an unlimited number of contexts without additional training. In contrast, other baselines underperform the original models in most tasks, highlighting their limitations in real-world scenarios.

4.2 Few-shot Learning

Setup. We evaluate APE on GSM8K (8-shot) [5], TriviaQA (5-shot) [14], and MMLU (5-shot) [12]. Baselines include parallel and sequential encoding with varying numbers of shots.

Results. In Figure 5, APE significantly surpasses parallel encoding with average improvements of 15.4% on GSM8K, 4.7% on TriviaQA, and 3.5% on MMLU. Moreover, APE achieves better performance than half-shot sequential encoding in 8/12 settings and preserve 93% accuracy comparing to the full-shot sequential encoding with using similar context length to the one-shot baseline.

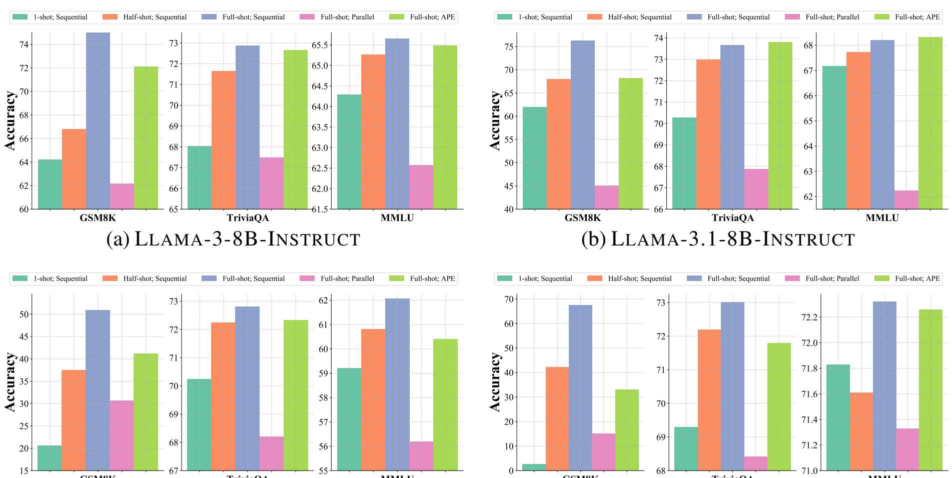
4.3 Efficiency Evaluation

Setup. We measured the prefilling time and total generation time for sequential encoding and APE on Llama-3.1-8B-Instruct [8] using VLLM [15]. Our evaluation were conducted on an H100. The query and generation lengths were 256 tokens, while context varied across 2K, 8K, 32K, 128K, and 512K.

Results. Table 2 demonstrates that our method can accelerate inference up to 756$\times$ in long-context scenarios, where the prefilling time dominates the overall process. For a 512K-token prompt with

Table 1: Performance of various methods on different models with LongBench [3] samples exceeding 8K tokens. Markers ○ and ● refer to training-required and inference-only methods.

Methods	Length	HotpotQA	2WikiMQA	MuSiQue	MultiNews	Avg.
LLAMA-3-8B-INSTRUCT	8K	44.45	23.63	20.91	23.26	28.07
○ LLMLingua2	40K	40.16	24.72	20.85	21.34	26.77
● StreamingLLM	∞	32.76	20.12	17.32	21.49	22.92
○ Long-context FT	262K	15.89	10.49	8.74	**24.28**	14.85
● PCW	∞	37.37	24.47	11.59	20.02	23.36
● APE	∞	**44.68**	**25.48**	**22.85**	22.93	**28.99**
LLAMA-2-7B-CHAT	4K	24.15	23.12	7.92	23.17	19.59
○ LLMLingua2	20K	27.79	19.35	11.07	20.68	19.72
● StreamingLLM	∞	14.74	14.17	3.99	18.93	12.96
○ Long-context FT	32K	13.39	7.35	7.41	22.28	12.61
○ CEPE(D)	∞	26.25	18.08	8.78	16.02	17.28
● PCW	∞	25.80	20.01	7.28	21.64	18.68
● APE	∞	**34.59**	**23.25**	**9.37**	21.97	**22.30**
GEMMA-2-9B-IT	8K	43.38	31.27	20.81	23.16	29.66
○ LLMLingua2	40K	48.63	43.37	23.87	18.73	33.65
● StreamingLLM	∞	32.61	27.9	17.39	20.16	24.52
● PCW	∞	47.06	34.04	22.60	20.75	31.12
● APE	∞	**51.16**	**37.10**	**28.01**	22.89	**34.79**



(a) LLAMA-3-8B-INSTRUCT (b) LLAMA-3.1-8B-INSTRUCT

(c) MISTRAL-7B-INSTRUCT-V0.3 (d) GEMMA-2-9B-IT

Figure 5: Performance comparison of APE, parallel encoding, and sequential encoding on ICL tasks.

Table 2: Latency on H100 GPU: [prefill / generation time (ms)]. The gray text in brackets is batch size.

Methods	2K	8K	32K	128K	512K
Sequential Encoding (1)	67/1983	269/2484	1623/4106	16063/19661	250926/259798
APE (1)	6/1922	10/2225	36/2483	91/3689	332/9204
Sequential Encoding (4)	275/2288	1097/3601	6502/10091	64807/9185	OOM
APE (4)	29/2042	63/2576	83/3672	108/9293	OOM

256 tokens generated, prefilling occupies 97% of the total generation time. Moreover, as context length and batch size increase, prefilling time rises significantly faster than decoding time. This is because prefilling is computation-bound, making it less susceptible to acceleration through I/O optimizations.

5 Conclusion

In summary, the work explores the potential of parallel encoding, which pre-cache the context for fast inference and reuse positions for extended context but leads to worse performance. To address this, we propose APE to enable accurate, fast, and long context-augmented generation without requiring additional fine-tuning. APE achieves this by aligning the attention weight distribution of parallel encoding with sequential encoding via three steps: shared prefix, scaling factor, and adaptive temperature. Our method improves both efficiency and performance in RAG and ICL scenarios.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Stephanie Chan, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, et al. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*, 2024.

[3] 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. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

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

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[6] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[7] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

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

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

[10] AI Gradient. Llama-3-8b-instruct-262k. 2024.

[11] Aman Gupta, Anup Shirgaonkar, Angels de Luis Balaguer, Bruno Silva, Daniel Holstein, Dawei Li, Jennifer Marsman, Leonardo O Nunes, Mahsa Rouzbahman, Morris Sharp, et al. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*, 2024.

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

[13] Ziyan Jiang, Xueguang Ma, and Wenhu Chen. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*, 2024.

[14] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

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

[16] Zhenyu Li, Yike Zhang, Tengyu Pan, Yutao Sun, Zhichao Duan, Junjie Fang, Rong Han, Zixuan Wang, and Jianyong Wang. Focusllm: Scaling llm's context by parallel decoding. *arXiv preprint arXiv:2408.11745*, 2024.

[17] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*, 2024.

[18] Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. *arXiv preprint arXiv:2212.10947*, 2022.

[19] East Sun, Yan Wang, and Lan Tian. Block-attention for efficient rag. 2024.

[20] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[21] AI Together. Llama-2-7b-32k-instruct. 2023.

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

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

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

[25] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, et al. Crag–comprehensive rag benchmark. *arXiv preprint arXiv:2406.04744*, 2024.

[26] Howard Yen, Tianyu Gao, and Danqi Chen. Long-context language modeling with parallel context encoding. *arXiv preprint arXiv:2402.16617*, 2024.

A Appendix

In Appendix, we present analyses to answer the following research questions: **RQ1**: Does each alignment process in APE function effectively? **RQ3**: Can APE generalize to more general scenarios featuring single long-context input? **RQ4**: Can APE improve performance for real-world RAG tasks?

A.1 How does each step in APE contribute to the performance?

In Table 3, we conduct an ablation study to examine each alignment process in APE, including the shared prefix (S_P), scaling factor (s), and attention temperature (T). We present results averaged across the four models evaluated in Figure 5. Our findings indicate that incorporating each of these components can consistently enhance performance across all tasks, with average improvements of 5.19%, 0.59%, and 2.07% , respectively. Among them, adding the scaling factor yields minimal performance gains without the complementary effect of attention temperature.

Table 3: Ablation study of APE on three ICL tasks. S_P: shared prefix, s: scaling factor, and T: attention temperature.

S_P	s	T	GSM8K	TriviaQA	MMLU
			38.25%	67.99%	63.09%
✓			50.42%	70.76%	63.70%
✓	✓		51.15%	71.03%	64.49%
✓	✓	✓	**53.62%**	**72.64%**	**66.62%**

A.2 Can APE work with single, continuous long context?

Table 4 examines the effectiveness of APE when processing a single long context input for the LLAMA-3-8B-INSTRUCT model on LongBench [3]. To accommodate the long context within our APE, we spilt it into multiple segments, each containing fewer than 7,500 tokens. The results demonstrate that APE enhances performance on 6 tasks, with the exception of two code completion tasks. This limitation arises from the disruption of long-range dependencies within the original context, leading to performance degradation in tasks that heavily rely on these contextual relationships.

Table 4: Performance comparison between the LLAMA-3-8B-INSTRUCT model with and without APE on LongBench [3] . All eight tasks involve single, continuous long-context inputs.

Methods	NarratQA	Qasper	MultiFQA	GovReport	SAMSum	LCC	RepoBench-P
LLAMA-3-8B-INSTRUCT	18.74	26.11	42.91	27.98	42.46	**53.10**	**38.83**
+ APE	**21.52**	**38.55**	**47.13**	**28.67**	**43.31**	32.89	23.45

A.3 Can APE work in real-world RAG applications?

In Table 5, we evaluate APE's performance in real-world RAG scenarios using the CRAG benchmark [25]. Task 1 augments the model with several webpages, while Task 2 provides an additional knowledge graph. By incorporating significantly more external data during generation, APE consistently outperforms sequential encoding that have limited context sizes. Moreover, the improvement in Task 2 further shows our method's effectiveness in merging text from multiple sources.

Table 5: Performance comparison using LLAMA-3-8B-INSTRUCT on CRAG.

Task	Model	Accuracy	Hallucination	Missing	Score_a
LLM only	LLAMA-3-8B-INSTRUCT	22.14	48.97	28.90	-26.83
Task 1	LLAMA-3-8B-INSTRUCT	23.28	29.49	47.22	-6.21
	+APE	**25.53**	**21.30**	**37.93**	**-0.41**
Task 2	LLAMA-3-8B-INSTRUCT	24.46	28.38	47.15	-3.92
	+APE	**27.04**	**18.74**	**37.32**	**2.16**