Anchor-based Large Language Models

Jianhui Pang[1*] **Fanghua Ye**[2*] **Derek Fai Wong**[1†] **Xin He**[3]
Wanshun Chen[3] **Longyue Wang**[3†]
[1]University of Macau [2]University College London [3]Tencent AI Lab
nlp2ct.pangjh3@gmail.com, fanghua.ye.19@ucl.ac.uk, derekfw@um.edu.mo
{shaynechen, kleinhe, vinnylywang}@tencent.com

Abstract

Large language models (LLMs) predominantly employ decoder-only transformer architectures, necessitating the retention of keys/values information for historical tokens to provide contextual information and avoid redundant computation. However, the substantial size and parameter volume of these LLMs require massive GPU memory. This memory demand increases with the length of the input text, leading to an urgent need for more efficient methods of information storage and processing. This study introduces Anchor-based LLMs (AnLLMs), which utilize an innovative anchor-based self-attention network (AnSAN) and also an anchor-based inference strategy. This approach enables LLMs to compress sequence information into an anchor token, reducing the keys/values cache and enhancing inference efficiency. Experiments on question-answering benchmarks reveal that AnLLMs maintain similar accuracy levels while achieving up to 99% keys/values cache reduction and up to 3.5 times faster inference. Despite a minor compromise in accuracy, the substantial enhancements of AnLLMs employing the AnSAN technique in resource utilization and computational efficiency underscore their potential for practical LLM applications.[1]

1 Introduction

Large language models (LLMs) primarily utilize decoder-only transformer architectures, which necessitate caching keys/values information for historical tokens during the auto-regressive inference to supply contextual information and avoid redundant computation (Wei et al., 2022; Touvron et al., 2023a; OpenAI, 2023; Touvron et al., 2023b). However, due to their immense size and high parameter count, a considerable amount of GPU memory is required for loading. Furthermore, as the length of input text grows, storing keys/values caches requires more and more GPU memory, as evidenced in in-context learning, complex instructions, and extended conversations (Dong et al., 2022; Jiang et al., 2023; Wang et al., 2023a), which is not conducive to scenarios with limited computational resources. An alternative approach entails recalculating these extensive inputs, which, however, results in increased time overhead. Therefore, this study aims to *reduce the storage demand for keys/values caches during the inference phase of LLMs, improving the memory efficiency and, consequently, accelerating the inference speed as well.*

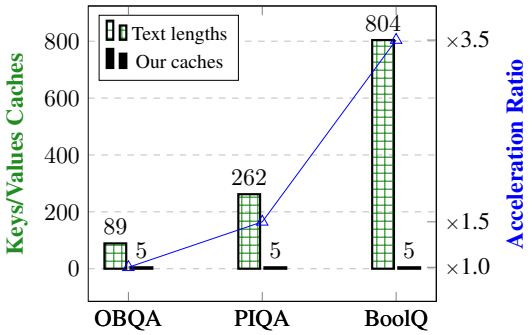

Figure 1: Keys/Values Caches and Inference Acceleration Ratio of Ours in OBQA, PIQA, and BoolQ Tasks with Five-Shot Demonstrations. The bars indicate Keys/Values cache and text length, while the curve shows the Inference Acceleration Ratio. As text length increases, our method achieves up to 99% reduction in Keys/Values Caches compared to traditional approaches. Moreover, caching prefix texts enhances inference efficiency by 3.5 times over non-caching methods.

In a recent study, Wang et al. (2023a) demonstrate that label words in prefix demonstrations can act as anchors during inference, providing an effective context compression approach for improving inference efficiency in in-context learning. However, in practical applications, not all prefix inputs or demonstrations contain label words suitable for

[*]Work was done when Jianhui Pang and Fanghua Ye were interning at Tencent AI Lab.
[†]Corresponding Authors.
[1]Our code and models are publicly available at: https://github.com/pangjh3/AnLLM.

Findings of the Association for Computational Linguistics ACL 2024, pages 4958–4976

compressing information, making the reliance on label words a less universal approach for text information compression. Additionally, Pang et al. (2024b) observe that LLMs tend to attend to only a few, yet consistent, prefix tokens during inference. However, the specific tokens utilized are often unpredictable and uncontrollable. These observations raise an intriguing question: *do natural language texts contain anchor points that compress the overall semantic information of sequences?* In this context, previous studies on sequence embeddings have shown that the hidden state of a special token in neural network models can encapsulate semantic information (Baudiš et al., 2016; Devlin et al., 2018). Furthermore, contemporary LLMs typically utilize the causal self-attention mechanism during both training and inference phases (Touvron et al., 2023a,b), attending on each prior token. This suggests that the final token in a sequence may be better suited to serve as a natural information compression point compared to other tokens, as they cannot observe future tokens. Therefore, a methodical approach that identifies and exploits sequence anchor tokens in a dependable and controllable manner is essential for compressing sequence information, effectively reducing keys/values caches, and improving inference efficiency for LLMs.

To this end, we propose novel **An**chor-based **L**arge **L**anguage **M**odels (AnLLMs), equipped with an innovative anchor-based self-attention network (AnSAN) and an anchor-based inference strategy. The AnSAN is designed to compel the models to compress sequence information into the anchor token (the last token in our implementation) during the training process, with the aid of anchor-based attention masks. During inference, the anchor-based inference strategy retains the keys/values caches of anchor tokens, which have aggregated the entire sequence information, and discards those of non-anchor tokens, thereby reducing memory demands. Specifically, the anchor-based attention masks for AnSAN serve two objectives: 1) to ensure anchor tokens attend exclusively to tokens within the same sequence, preventing attention to other sequences, and 2) to direct non-anchor tokens' attention to previous sequence anchors, blocking the other non-anchor tokens from previous sequences. It is noteworthy that the technique of anchor-based attention bears similarities to the principles underlying sparse attention (Child et al., 2019). However, unlike the existing research that employs sparse attention to extend the context length of LLMs (Chen et al., 2023; Ratner et al., 2023), our method focuses on continually pre-training the model to compress sequence information into the anchor token.

In our implementation, we utilize the publicly available RedPajama datasets (Computer, 2023) to continuously pre-train the open-source Llama2 models (Touvron et al., 2023b), resulting in AnLLMs that incorporate our proposed anchor-based attention mechanism. Experimental results on question answering benchmarks, as depicted in Figure 1, reveal that our method achieves up to a 99% reduction in keys/values caches and up to a 3.5-fold increase in inference acceleration ratios, while maintaining comparable accuracy to the original model. Despite a minor decrease in accuracy (within 1.5%), these findings underscore the significant improvements in computational efficiency and memory utilization offered by our method.

2 Related Work

Our research is inspired by the recent investigation into the understanding of in-context learning (ICL) within LLMs by Wang et al. (2023a). In their study, the authors delve into the underlying mechanisms of ICL, emphasizing the influence of label words in demonstration examples on information flow. They reveal that these label words serve as anchors, wherein semantic information converges into these anchors during inference, subsequently directing the LLMs' final predictions. Motivated by their findings, our objective is to extend this feature to natural language modeling by guiding sequence information compression into manually designed anchor tokens, rather than solely relying on label words. This is crucial because natural language texts may not always contain an explicit label.

The most relevant method to our approach in the existing literature is the learning to compress prompts with gist tokens (Mu et al., 2023). Their approach centers around compressing task-specific prompts by fine-tuning the model using the proposed gist masking, thereby enforcing prompt compression. However, there are several crucial divergences between our study and theirs. Unlike their focus on compressing a task prompt, our objective lies in training the LLM to condense sequence information into the anchor tokens. Consequently, our approach can be universally applied to a range of tasks without requiring task-specific training, a feature not shared by gist tokens, as the anchor tokens are seamlessly incorporated into the model's lan-

guage modeling. Furthermore, our anchor-based attention masks account for information compression within a sequence and information interaction between sequences, thus extending beyond the mere compression of task prompts.

On the other hand, FlashAttention (Dao et al., 2022) and PagedAttention (Kwon et al., 2023) both present memory-efficient attention mechanisms for LLMs. While they focus on optimizing attention computation and subdividing attention processing, our proposed method offers a distinct approach that specifically targets the compression of sequence information into anchor tokens, making it orthogonal to these existing works.

3 Anchor-based Large Language Models

3.1 Background

Transformers. LLMs are primarily realized as decoder-only transformers (Vaswani et al., 2017; Touvron et al., 2023a,b), incorporating an input embedding layer and multiple decoder layers. Each layer contains a self-attention network and a feed-forward network with normalization modules. Crucially, causal attention masks are employed, allowing tokens to attend only to preceding ones.

Self-Attention Networks. Typically for decoder-only LLMs like Llama2 (Touvron et al., 2023b), self-attention networks (SANs) map queries Q, keys K, and values V into an output, as delineated in the following equations,

$$\text{SAN}(Q,K,V) = \text{Softmax}(Q,K)V, \tag{1}$$

$$\text{Softmax}(Q,K)_{i,j} = \frac{M_{i,j}\exp(Q_i K_j^T)}{\Sigma_k M_{i,k}\exp(Q_i K_k^T)}, \tag{2}$$

$$M_{i,j} = \begin{cases} 1, & \text{if } i \geq j \\ 0, & \text{else} \end{cases}, \tag{3}$$

where M denotes an $L \times L$ masking matrix, facilitating the current i-th token to attend to only preceding tokens whilst disregarding subsequent tokens during the training and inference phases.

Keys/Values Caches. In the application of LLMs, the keys/values caches increase with lengthy prefix texts and continuously generated tokens during the inference phase, such as in question-answering (Saad-Falcon et al., 2023), text summarization (Basyal and Sanghvi, 2023), and machine translation (Pang et al., 2024b). The key and value matrices associated with tokens of prefix inputs are cached to avoid recomputation and expedite subsequent token prediction (Radford et al., 2019). Additionally, the model generates the output token-by-token in the real-time inference process, which requires more cache memory to store the newly generated sequence. Therefore, addressing the challenges arising from the ever-expanding texts is crucial for enhancing the efficiency of LLM inference.

3.2 Anchor-based Self-Attention Networks

Given an input text with n consecutive sequences, $P = \{S_1, S_2, ..., S_n\}$, their associated anchor tokens are the last tokens that represented as $A = \{a_1, a_2, ..., a_n\}$. The primary objective of AnSAN is to encapsulate the information of a sequence into its anchor token, with the anchor hidden states representing the comprehensive semantic information. In this manner, an AnLLM equipped with AnSAN generates subsequent tokens based on the keys/values caches of preceding tokens within the current sequence and the keys/values caches of anchor tokens from previous sequences.

Anchor-based Attention Masks. To accomplish this, we devise anchor-based attention masks, as illustrated in Figure 2. Assuming that the current token in the sequence is a non-anchor token, we allow attention towards previous non-anchor tokens within the same sequence and anchor tokens from preceding sequences, while blocking attention towards non-anchor tokens from previous sequences. This approach ensures that non-anchor tokens can only access information from anchor tokens in previous sequences and the current sequence's information. Conversely, when the current token is an anchor token, which is the last token in the sequence, we exclusively permit its attention towards previous non-anchor tokens within the same sequence, blocking all other attention. This constraint forces the anchor token to aggregate information solely from its current sequence. Consequently, we replace Eq. (3) with anchor-based attention masks in Eq. (4) to determine the mask of the i-th token in the input text concerning the j-th token (assuming that the i-th token belongs to the k-th sequence).

$$M_{i,j} = \begin{cases} 0, & \text{if } ((w_i, w_j) \notin A) \wedge (w_j \in S_{\leq k-1}) \\ 0, & \text{else if } (w_i \in A) \wedge (w_j \in S_{\leq k-1}) \\ 1, & \text{else if } i \geq j \\ 0, & \text{else} \end{cases}, \tag{4}$$

where $S_{\leq k-1}$ represents previous $k-1$ sequences. The number 0 denotes blocking attention, whereas the number 1 indicates the opposite.

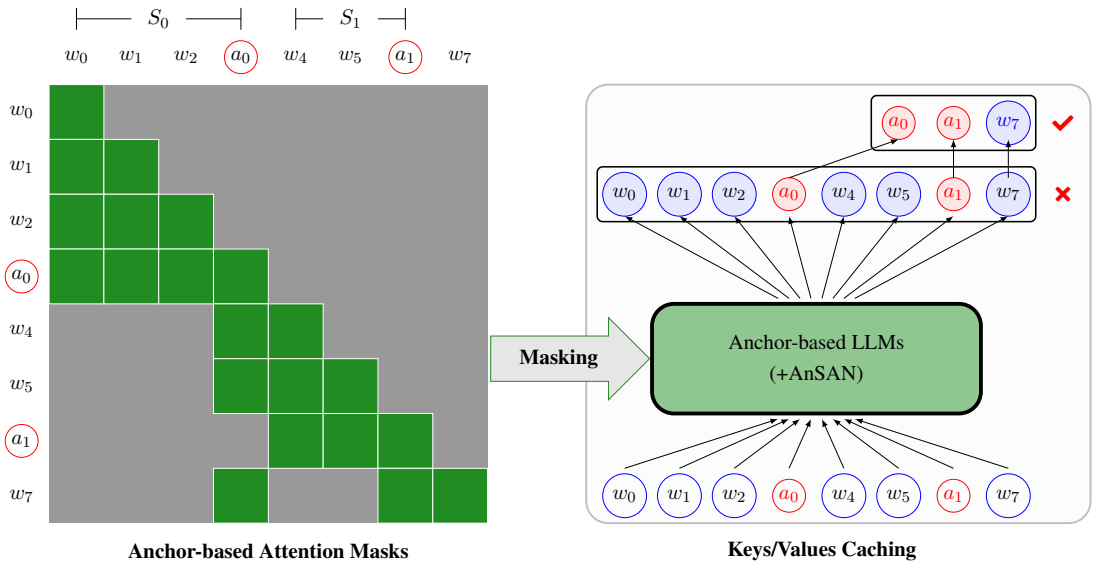

Figure 2: Anchor-based Attention Masking and Efficient Caching in Anchor-based LLMs. On the left, the gray and green squares represent the masking and unmasking operations respectively, with the circled "a" symbols denoting the anchor tokens. On the right, the shaded circles depict keys/values caches. By employing anchor-based attention masking during training, we compel the model to compress sequence information into the anchor tokens. On the right, during inference, with the AnSAN technique, AnLLMs compress information into the anchor tokens and discard the previous remaining keys/values caches, thereby facilitating an efficient caching mechanism.

Anchor Token Selection. By implementing the AnSAN mechanism for training LLMs, we can compel the model to compress sequence information into the anchor token and generate new tokens based on the anchor token information from previous sequences and non-anchor token information from the current sequence.

The challenge now lies in selecting an appropriate anchor token. In our experiments, we propose two implementation methods: one using the endpoint as the anchor token, and the other appending a new token specifically as the anchor token.

3.3 Anchor-based Inference

By training the model to compress information into the anchor token of a natural language sequence, we can optimize the inference process by modifying the keys/values caching mechanism. Specifically, during inference, upon encountering an anchor token that condenses the comprehensive semantic information of preceding tokens in the current sequence, the model can reduce the keys/values caches by deleting the caches of non-anchor tokens within that sequence.

We introduce the inference method in Algorithm 1. The function "REDUCTION" in Line 1 is utilized to remove keys/values caches when the model processes prefix texts in Line 10 or generates an anchor token during the prediction of the next token in Line 16. This approach aims to reduce the keys/values caches for both prefix tokens and generated outputs during real-time inference.

Algorithm 1 Anchor-based Inference

Require: Anchor-based LLM Θ, prefix text P with anchor tokens, keys/values cache list $\mathcal{C}$, predicted token w_{new};
Output: Generated text $\mathcal{T}$;
1: **function** REDUCTION($\mathcal{C}$)
2: $j \leftarrow$ last anchor index in $\mathcal{C}$;
3: $\mathcal{C} \leftarrow \{c \in \mathcal{C} \mid \text{index}(c) \geq j \text{ or } c \text{ is anchor}\}$;
4: **return** $\mathcal{C}$.
5: **end function**
6: Initialize $\mathcal{T}, \mathcal{C}$ as empty lists;
7: $\mathcal{M} \leftarrow$ GetMasks($P, \mathcal{C}$) using Eq. (4);
8: Update $w_{new}, \mathcal{C}$ using Forward(P;$\mathcal{M}, \mathcal{C}, \Theta$);
9: Append w_{new} to $\mathcal{T}$;
10: $\mathcal{C} \leftarrow$ Reduction($\mathcal{C}$);
11: **while** w_{new} is not [eos] **do**
12: $\mathcal{M} \leftarrow$ GetMasks($w_{new}, \mathcal{C}$) using Eq. (4);
13: Update $w_{new}, \mathcal{C}$ using Forward(w_{new};$\mathcal{M}, \mathcal{C}, \Theta$);
14: Append w_{new} to $\mathcal{T}$;
15: **if** w_{new} is the anchor token **then**
16: $\mathcal{C} \leftarrow$ Reduction($\mathcal{C}$);
17: **end if**
18: **end while**
19: **return** $\mathcal{T}$.

4 Experimental Setups

In this section, we first detail AnLLM's implementation, then outline the training procedure and model perplexity. Finally, we introduce the evaluation datasets and metrics.

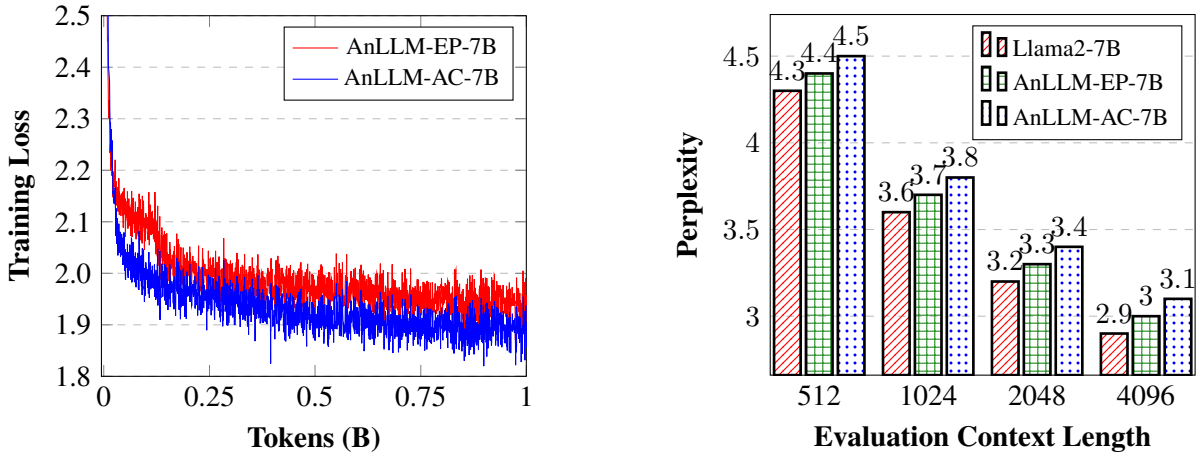

Figure 3: Training Process and Perplexity Evaluation of the Anchor-based Large Language Model.

4.1 Our Implementation

Llama2-7b (Touvron et al., 2023b) is adopted as the base model in our experiments, which is an open-source and English-centric LLM. In accordance with the principles outlined in Section 3, we present our implementations here. The crux is to identify which tokens in a sequence can be considered anchor tokens. In light of this, we describe two implementation strategies: one employs the endpoints directly, and the other involves appending a new token at the end of a sequence to serve as the anchor token. The details are as follows:

- **AnLLM-EP.** This approach uses punctuation marks within the sequence as anchor tokens. Punctuation marks, such as commas, periods, and question marks, are viewed as semantic boundaries within a sequence. As such, they can serve as anchor tokens in AnLLM. In our experiments of AnLLM-EP, we use the endpoint in English as the anchor tokens.
- **AnLLM-AC.** This strategy entails the introduction of a new token to act as the sequence anchor. In our implementation, we designate <AC> as the new token and initialize its embedding using the mean value of the embedding matrix. For training data, we use the sentence tokenizer from the NLTK package to split texts into sentences, appending <AC> at the end of each sentence as the anchor token.[2] During inference, <AC> tokens can be flexibly added to the text based on user requirements, such as adding one anchor for each demonstration, allowing for flexible and controllable sequence compression.

4.2 Data and Training Procedure

Considering that AnLLMs are expected to predict subsequent tokens within the context of keys/values hidden states of anchor tokens, this presents a significant challenge for existing open-source LLMs. To this end, by substituting the self-attention networks with anchor-based self-attention networks as detailed in Section 3.2, we continually pre-train the Llama2 model using a publicly available corpus.

Data. We employ the RedPajama-Data-1T-Sample dataset (Computer, 2023) for the continuous pre-training purpose.[3] This dataset comprises $850,000$ samples with approximately 1 billion tokens, which have been subjected to right truncation to fit the model context of $4,096$.

Training Procedure. We train each model via the next token prediction objective on the dataset for one epoch, with a batch size of 512. The learning rate is set to 0.00002 and constant after a linear warmup with 20 update steps. The AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ is adopted as the gradient backtrack propagation optimizer. All the training procedures are conducted with four $8\times$ A100 GPU machines with 40GB GPU Memory.

Training Loss and Perplexity. The left-hand side of Figure 3 depicts the training loss associated with our models. The loss curves for AnLLM-EP and AnLLM-AC consistently decline to approximately 1.9, with AnLLM-AC achieving a lower loss. This observation suggests that continually pre-training an LLM using anchor-based attention

[2] https://www.nltk.org/api/nltk.tokenize.punkt.html

[3] https://huggingface.co/datasets/togethercomputer/RedPajama-Data-1T-Sample

masks is indeed viable, enabling the LLM to effectively learn the process of compressing sequence information into anchor tokens.

The right-hand side of Figure 3 displays the perplexity evaluation of the models with varying context lengths. Full attention is utilized to assess the language modeling capabilities of all models. Following the settings of Chen et al. (2023), the perplexity is evaluated on the test samples of the Proof-Pile datasets (Rae et al., 2020). The results demonstrate that both AnLLM-EP and AnLLM-AC models maintain a promising performance, exhibiting language modeling capacity comparable to the base model, Llama2-7B. Moreover, this finding suggests that AnLLMs are compatible with full attention, as indicated by minimal perplexity decline.

4.3 Evaluation

In our investigation, we employ a diverse collection of benchmarks with varying text lengths to evaluate our outcomes, including OpenBookQA (OBQA) (Mihaylov et al., 2018), WinoGrande (WG) (Sakaguchi et al., 2021), ARC-easy (ARC-e) and ARC-challenge (ARC-c) (Clark et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (HS) (Zellers et al., 2019), SCIQ (Welbl et al., 2017), and BoolQ (Clark et al., 2019). These benchmarks provide a comprehensive evaluation of various aspects, including reasoning, comprehension, understanding of the physical world, and predicting future events. Importantly, they cover texts of varying lengths, facilitating a thorough assessment of our model's performance across diverse tasks and text complexities, ranging from shorter input contexts in OBQA to longer texts in BoolQ. To measure the precision and efficiency of our models, we evaluate them across three dimensions using three distinct metrics for both zero-shot and five-shot settings. For AnLLM-AC in the five-shot setting, we incorporate the anchor token <AC> at the end of each demonstration.

- **Accuracy (*Acc*).** This conventional metric is utilized to gauge the prediction accuracy of models. In accordance with previous studies (Gao et al., 2023), we choose the options with the highest probabilities as predictions and calculate accuracy using the gold-standard labels.
- **Keys/Values Caches Reduction ($\mathcal{C}_{\Downarrow}$).** In the context of the five-shot evaluation, the demonstrations can be cached in GPU memory for subsequent reuse. Nevertheless, extended demonstrations may require increased memory consumption. This metric is designed to assess the memory efficiency of the AnSAN technique.
- **Inference Acceleration Ratio ($\mathcal{T}_{\Uparrow}$).** Similar to Wang et al. (2023a), capitalizing on the cached keys/values, we present the inference acceleration ratio, which serves as an indicator of the inference efficiency of the AnSAN technique.

Note that we first report full attention inference results for all models, then present results with the AnSAN method (+AnSAN) applied, compressing sequence information into anchor tokens.

5 Experimental Results

As evident from the results presented in Table 1, both the AnLLM-AC and AnLLM-EP models demonstrate promising accuracy, comparable to that of the base model, while simultaneously improving memory and inference efficiency.

Accuracy (*Acc*). The proposed AnLLM-EP and AnLLM-AC models exhibit commendable accuracy across various benchmarks.

In the zero-shot setting, with full attention, AnLLM-EP and AnLLM-AC achieve average accuracies of 64.6% and 65.1%, respectively, comparable to Llama2-7B's 65.8% accuracy. This suggests that training with integrated anchor tokens barely affects the model capacity, emphasizing the robustness of LLMs. Furthermore, our models excel in OBQA, PIQA, and SCIQ tasks.

In the five-shot setting, with five prior examples, AnLLM-EP and AnLLM-AC maintain dependable performance using full attention. When implementing the AnSAN technique, a slight accuracy decline across all models is observed. This is expected, as AnSAN, designed for memory efficiency, necessitates token removal, potentially leading to information loss. The degradation in BoolQ is most pronounced, which contains the longest demonstration tasks, indicating that the longer the text, the greater the information loss after compression. However, the average accuracy reduction is minimal, approximately 1.5%, suggesting that AnSAN effectively balances memory-saving and model performance.

Keys/Values Cache Reduction ($\mathcal{C}_{\Downarrow}$). The size of the keys/values cache is a critical factor in the practical implementation of LLMs, particularly concerning memory efficiency and computational resources. In this respect, the AnLLM-EP and AnLLM-AC models offer significant advantages.

	OBQA	WG	ARC-e	ARC-c	PIQA	HS	SCIQ	BoolQ	AVG.
Llama2-7B	31.4	69.1	76.3	43.4	78.1	57.1	93.7	77.7	65.8
AnLLM-EP	33.2	68.0	73.4	40.8	77.8	55.0	94.4	74.4	64.6
AnLLM-AC	31.6	68.5	74.4	42.5	78.3	54.7	93.8	77.0	65.1

(a) The Zero-Shot Performance.

		OBQA	WG	ARC-e	ARC-c	PIQA	HS	SCIQ	BoolQ	AVG.
	L_d	89	133	145	209	262	426	603	804	334
	L_x	18	26	36	42	42	90	130	169	69
Llama2-7B	*Acc*	37.2	73.7	79.8	50.0	78.7	58.3	96.8	78.4	69.1
+AnSAN	*Acc*	34.6	68.6	62.6	35.8	68.3	30.8	65.7	50.8	52.1
AnLLM-EP	*Acc*	36.8	71.0	79.4	49.4	78.1	55.3	96.6	75.6	67.8
+AnSAN	*Acc*	36.2	68.0	76.7	45.6	78.2	52.6	93.1	74.0	65.6
	L_{kv}	89	8	5	30	9	25	50	43	32
	$\mathcal{C}_{\Downarrow}$	–0%	–94%	–97%	–86%	–97%	–94%	–92%	–95%	–90%
	$\mathcal{T}_{\Uparrow}$	×1.0	×1.0	×1.0	×1.2	×1.4	×2.1	×2.6	×3.5	×1.7
AnLLM-AC	*Acc*	37.2	72.3	79.8	49.0	78.6	56.9	96.8	77.5	68.5
+AnSAN	*Acc*	35.6	70.6	79.2	47.9	78.7	55.6	95.7	76.6	67.5
	L_{kv}	5	5	5	5	5	5	5	5	5
	$\mathcal{C}_{\Downarrow}$	–94%	–96%	–97%	–98%	–98%	–99%	–99%	–99%	–99%
	$\mathcal{T}_{\Uparrow}$	×1.0	×1.0	×1.1	×1.2	×1.5	×2.0	×2.6	×3.5	×1.7

(b) The Five-Shot Performance.

Table 1: Accuracy and Efficiency of LLMs on Question Answering Benchmarks. $\mathcal{C}_{\Downarrow}$ represents the reduction in keys/values cache size, while $\mathcal{T}_{\Uparrow}$ denotes the inference acceleration ratio during testing. *Acc* stands for Accuracy. L_{kv} represents the length of the keys/values cache. L_d and L_x denote the lengths of in-context learning demonstrations and input queries, respectively. Our methods effectively reduce cache sizes and boost inference efficiency.

By adopting the AnSAN, these models are designed to dramatically reduce the keys/values cache size during inference. As shown in Table 1, these models achieve remarkable reductions in cache size. Specifically, the average reduction percentages are around 90% for AnLLM-EP and an impressive 99% for AnLLM-AC. This is a substantial improvement compared to conventional approaches, which typically necessitate large cache sizes to store keys/values. These reductions in cache size translate to considerable savings in memory and computational resources, rendering these models highly efficient for practical applications.

Inference Acceleration Ratio ($\mathcal{T}_{\Uparrow}$). The inference acceleration ratio serves as a crucial metric reflecting the model's efficiency during the testing phase. By incorporating anchor tokens into natural language texts, we can repurpose the hidden states of anchor tokens as keys/values caches in the demonstrations, and then adopt an inference strategy as suggested by Wang et al. (2023a). In this scenario, both the AnLLM-EP and AnLLM-AC models demonstrate significant improvements.

Specifically, in the five-shot testing, both AnLLM-EP and AnLLM-AC models attain an average inference acceleration ratio of approximately 1.7 times. This represents a considerable advancement over the conventional non-caching method, which typically necessitates prolonged processing times due to the large number of tokens involved. As L_d increases, reaching up to 3.5 times in the BoolQ task, the acceleration ratios also escalate, corroborating the findings of Wang et al. (2023a). This enhancement in processing speed leads to increased efficiency, making these models particularly apt for scenarios with limited resources.

The AnLLM-EP and AnLLM-AC models exhibit remarkable performance in natural language understanding benchmarks, effectively balancing accuracy, memory efficiency, and time acceleration. The incorporation of anchor tokens into AnLLMs, along with the utilization

of the AnSAN technique for reducing keys/values cache size, allows these models to maintain performance on par while significantly improving memory efficiency and inference speed. The equilibrium achieved between model performance and computational efficiency is noteworthy and opens up new possibilities for the advancement of LLMs.

6 Analysis

To further elucidate our method's insights, we conduct a natural language generation experiment with the German-to-English (De2En) translation task. We evaluate the models using COMET-DA (Rei et al., 2022), indicating translation quality, and the Keys/Values Cache Reduction $\mathcal{C}_{\Downarrow}$ metric, denoting memory efficiency as previously described. In line with previous findings, AnLLMs accept a minor accuracy trade-off (about 3 COMET-DA points) for enhanced memory efficiency. All LLMs are fine-tuned on the Alpaca dataset, combined with the newstest2017-2020 datasets, following Jiao et al. (2023). Results are presented in Table 2.

Model	Src Cache	De2En	MaxKV	$\mathcal{C}_{\Downarrow}$
Llama2-7b	✓	83.1	220	0%
AnLLM-EP	✓	81.6	220	0%
+AnSAN	✗	78.5	50	77%
	✓	80.3	124	44%
AnLLM-AC	✓	82.4	220	0%
+AnSAN	✗	78.0	35	84%
	✓	80.0	125	43%

Table 2: COMET-DA Scores and Keys/Values Cahces for the WMT23 German-to-English (De2En) Translation Task. The term "Src Cach" denotes retaining source sentence hidden states in Keys/Values Caches, while "MaxKV" refers to the average maximum keys/values length during inference.

6.1 Compatibility and Flexibility of Full Attention and Anchor-based Attention

The results offer significant insight into the interplay between anchor-based attention and full attention mechanisms in the De2En translation task. Since source sentences are vital in translation tasks, applying full attention to them is crucial for maintaining model performance. Thus, retaining the source sentence keys/values caches is expected to enhance AnLLM performance when implementing the AnSAN technique. Specifically, when combining full attention with the AnSAN method, both AnLLM-EP and AnLLM-AC achieve approximately 80.0 COMET-DAE scores, comparable to other models using full attention exclusively. This indicates that the AnSAN technique is compatible with the full attention mechanism. Consequently, our proposed models allow users to choose between full attention and anchor-based attention for input texts based on their needs, emphasizing the compatibility and flexibility of our models.

6.2 Effective Cache Reduction for Real-Time Inference with the AnSAN Technique

The results in Table 2 show that our reduction strategy effectively minimizes keys/values caches during real-time inference. Specifically, as indicated in Line 15 of Algorithm 1, when generating an anchor token (i.e., the endpoint or <AC> tokens), our AnSAN-equipped models execute the reduction function to minimize the current keys/values caches. When discarding source sentence caches, we achieve approximately 77% and 84% reduction for the AnLLM-EP and AnLLM-AC models, respectively, albeit with a low COMET-DA score. However, when retaining source sentence caches, we still reduce around 44% of caches for both models, achieving a COMET-DA score of approximately 80.0. These results confirm the effectiveness of our anchor-based inference strategy for practical real-time inference applications.

7 Ablation Studies

7.1 Impact of Anchor Positions

An intriguing question arises regarding the impact of anchor positions on model performance. In this section, we investigate the effects of varying anchor positions using the AnLLM-AC model, which enables us to modify the anchor position. Specifically, we employ the data settings from Section 3.2 and examine three position settings: the first compresses every 10 tokens, the second applies random compression, and the third compresses each demonstration, consistent with the setting in Table 1. For the second setting, an anchor token is randomly inserted after each token with a probability of 0.1. The experimental results are presented in Table 3.

Accordingly, we observe that the choice of anchor positions significantly affects the model's performance across various question-answering benchmarks. The "every-demonstration" setting consistently outperforms the other two settings, achieving the highest average accuracy of 67.5%. This suggests that strategically placing anchors at se-

Settings	OBQA	WG	ARC-e	ARC-c	PIQA	HS	SCIQ	BoolQ	AVG.
every-10-tokens	21.4	69.2	63.6	33.3	75.0	48.1	81.9	65.4	57.6
random-prob-0.1	21.6	69.9	64.8	34.8	75.5	48.1	80.0	67.4	57.8
every-demonstration	**35.6**	**70.6**	**79.2**	**47.9**	**78.7**	**55.6**	**95.7**	**76.6**	**67.5**

Table 3: Accuracy on Question Answering Benchmarks with Different Anchor Positions. All the experiments are conducted with the AnLLM-AC model in Table 1.

mantically meaningful positions, such as after each demonstration, can effectively enhance the model's ability to capture and utilize the information contained in the input texts.

In comparison, the "every-10-tokens" and "random-prob-0.1" settings yield lower average accuracies of 57.6% and 57.8%, respectively. These results indicate that compressing input texts at fixed intervals or randomly inserting anchor tokens may not be as effective in facilitating the model's understanding and reasoning processes. The suboptimal performance of these settings could be attributed to the potential loss of semantic coherence and structural information as a result of arbitrary compression or random anchor placement.

Overall, our ablation study highlights the importance of carefully selecting anchor positions in the AnLLM-AC model to maximize its performance on question-answering tasks. The superior performance of the "every-demonstration" setting demonstrates the benefit of aligning anchor positions with semantically meaningful boundaries in the input texts. Future research could explore more sophisticated strategies for anchor placement, taking into account the linguistic and contextual properties of the input data to further improve the model's performance on complex reasoning tasks.

7.2 Training from Scratch

To evaluate the language modeling capabilities of our anchor-based language model (AnLLM-EP), we perform a comparison with the standard Transformer model. This comparison involves the training of compact models from scratch, using the Wikitext-103 dataset.[4] Each model is configured with 18 layers, 4096 hidden states, and 16 heads. As shown in Figure 4, the AnLLM-EP model notably outperforms the standard Transformer model, achieving a lower perplexity of 32.81, compared to 36.57. This notable outcome suggests that the anchor-based training approach may enhance the effectiveness of language modeling tasks. In future research, it would be intriguing to further investigate the potential advantages of the anchor-based training strategy for training LLMs from scratch.

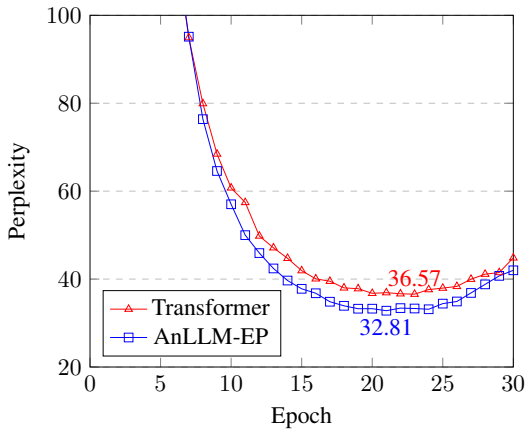

Figure 4: Perplexity Comparison Across Epochs for Small Standard Transformer and AnLLM-EP Models.

8 Conclusion

LLMs have emerged as a significant research area in the field of artificial intelligence. However, despite their exceptional performance across various natural language tasks, the practical application of these models is limited by their significant memory overhead and time efficiency. Implementing LLMs on resource-constrained devices, such as smartphones, poses a unique challenge. To address this issue, we propose anchor-based LLMs with the AnSAN technique. Our experiments demonstrate that by sacrificing a marginal 1.5% in precision, our approach saves 99% of keys/values cache memory while simultaneously improving inference speed by up to 3.5 times. Our methods' application in machine translation showcases their compatibility and flexibility, effectively enhancing memory efficiency for practical use. Our novel approach is practical, straightforward, flexible, and compatible with existing methods, paving the way for further adoption of LLMs in real-world applications.

[4] https://huggingface.co/datasets/iohadrubin/wikitext-103-raw-v1

Limitations

While our proposed AnLLMs demonstrate significant improvements in memory efficiency and inference acceleration, there are several limitations to consider:

1. **Accuracy Trade-off:** As observed in the experimental results, our method incurs a minor decrease in accuracy (within 1.5%) compared to the original model. This limitation stems from the information compression process, which may lead to information loss. Despite its minimal impact, this trade-off should be considered in practical applications. In future works, additional evaluation methods could further enrich our assessment (Ye et al., 2024; Wang et al., 2023c).

2. **Applicability to Various Tasks:** Our experiments primarily focus on question-answering benchmarks and machine translation tasks. The effectiveness of our method in other NLP tasks and domains remains to be thoroughly investigated. Future work will explore the applicability and performance of our method across a broader range of tasks and domains (Zhao et al., 2023; Wang et al., 2023b; Fang et al., 2023; Pang et al., 2024a; Wang et al., 2024; Lan et al., 2024; Zhan et al., 2024).

3. **Optimal Anchor Token Selection:** In our implementation, we chose the last token in a sequence as the anchor token. However, the optimal anchor token selection may vary across different tasks and domains. We encourage further analytical studies to explore the selection of anchor tokens.

4. **Scalability to Other LLMs:** We have applied our method to the open-source Llama2 models. It remains to be seen how our approach would perform when applied to other open-source LLMs, such as Falcon and Qwen (Almazrouei et al., 2023; Bai et al., 2023). Evaluating the effectiveness and scalability of our method on more extensive language models is an essential direction for future research.

Despite these limitations, our work presents a novel approach to enhancing memory efficiency and inference acceleration in LLMs. Future research efforts should address these limitations, refining our method and extending its applicability to a wider range of tasks and model architectures.

Ethics Statement

We place great importance on ethical considerations and rigorously adhere to the ACL Ethics Policy. In this paper, we propose an anchor-based large language model that reduces the Keys/Values cache size and enhances inference speed during the inference stage. The resources and methods employed in this paper are publicly accessible and have been extensively adopted by researchers in the field of large language models. We ensure that the findings and conclusions presented in this paper are reported accurately and objectively.

Acknowledgments

This work was supported in part by the Science and Technology Development Fund, Macau SAR (Grant No. FDCT/0070/2022/AMJ, the mainland China collaboration project, China Strategic Scientific and Technological Innovation Cooperation Project Grant No. 2022YFE0204900), the Science and Technology Development Fund, Macau SAR (Grant No. FDCT/060/2022/AFJ, the mainland China collaboration project, National Natural Science Foundation of China Grant No. 62261160648), the Multi-year Research Grant from the University of Macau (Grant No. MYRG-GRG2023-00006-FST-UMDF), and the Tencent AI Lab Rhino-Bird Gift Fund (Grant No. EF2023-00151-FST). This work was performed in part at SICC which is supported by SKL-IOTSC, and HPCC supported by ICTO of the University of Macau. We would like to thank the anonymous reviewers for their insightful comments.

References

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

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

Lochan Basyal and Mihir Sanghvi. 2023. Text summarization using large language models: A comparative study of mpt-7b-instruct, falcon-7b-instruct, and openai chat-gpt models. *arXiv preprint arXiv:2310.10449*.

Petr Baudiš, Silvestr Stanko, and Jan Šedivý. 2016. Joint learning of sentence embeddings for relevance and entailment. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 8–17, Berlin, Germany. Association for Computational Linguistics.

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

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *arXiv:2309.12307*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Tao Fang, Shu Yang, Kaixin Lan, Derek F. Wong, Jinpeng Hu, Lidia S. Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Wenxiang Jiao, Jen-tse Huang, Wenxuan Wang, Zhiwei He, Tian Liang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. ParroT: Translating during chat using large language models tuned with human translation and feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15009–15020, Singapore. Association for Computational Linguistics.

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

Kaixin Lan, Tao Fang, Derek F. Wong, Yabo Xu, Lidia S. Chao, and Cecilia G. Zhao. 2024. Focus: Forging originality through contrastive use in self-plagiarism for language models. In *Findings of ACL 2024*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

OpenAI. 2023. Gpt-4 technical report.

Jianhui Pang, Baosong Yang, Derek F. Wong, Dayiheng Liu, Xiangpeng Wei, Jun Xie, and Lidia S. Chao. 2024a. MoNMT: Modularly leveraging monolingual and bilingual knowledge for neural machine translation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11560–11573, Torino, Italia. ELRA and ICCL.

Jianhui Pang, Fanghua Ye, Longyue Wang, Dian Yu, Derek F Wong, Shuming Shi, and Zhaopeng Tu. 2024b. Salute the classic: Revisiting challenges of machine translation in the age of large language models. *arXiv preprint arXiv:2401.08350*.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6383–6402, Toronto, Canada. Association for Computational Linguistics.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, Ryan A Rossi, and Franck Dernoncourt. 2023. Pdftriage: Question answering over long, structured documents. *arXiv preprint arXiv:2309.08872*.

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

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

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

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

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Longyue Wang, Zefeng Du, Wenxiang Jiao, Chenyang Lyu, Jianhui Pang, Leyang Cui, Kaiqiang Song, Derek F Wong, Shuming Shi, and Zhaopeng Tu. 2024. Benchmarking and improving long-text translation with large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023b. Document-level machine translation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16646–16661, Singapore. Association for Computational Linguistics.

Longyue Wang, Zhaopeng Tu, Yan Gu, Siyou Liu, Dian Yu, Qingsong Ma, Chenyang Lyu, Liting Zhou, Chao-Hong Liu, Yufeng Ma, Weiyu Chen, Yvette Graham, Bonnie Webber, Philipp Koehn, Andy Way, Yulin Yuan, and Shuming Shi. 2023c. Findings of the WMT 2023 shared task on discourse-level literary translation: A fresh orb in the cosmos of LLMs. In *Proceedings of the Eighth Conference on Machine Translation*, pages 55–67, Singapore. Association for Computational Linguistics.

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

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek F Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. 2024. Benchmarking llms via uncertainty quantification. *arXiv preprint arXiv:2401.12794*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Runzhe Zhan, Xinyi Yang, Derek F Wong, Lidia S Chao, and Yue Zhang. 2024. Prefix text as a yarn: Eliciting non-english alignment in foundation language model. In *Findings of ACL 2024*.

Libo Zhao, Kai Fan, Wei Luo, Wu Jing, Shushu Wang, Ziqian Zeng, and Zhongqiang Huang. 2023. Adaptive policy with wait-k model for simultaneous translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4832, Singapore. Association for Computational Linguistics.

A More Experimental Results

A.1 Detailed Perpelixity Evaluation

In this section, we present a comprehensive analysis of the perplexity evaluation results, as illustrated in Table 4. The perplexity scores were calculated for various evaluation context lengths, ranging from 256 to 4096 tokens, utilizing the Proof-Pile datasets (Rae et al., 2020). The table compares the performance of Llama2-7B, AnLLM-AC, AnLLM-EP, and their corresponding variants incorporating the AnSAN technique. Our findings reveal that the AnSAN technique, on average, leads to a one-point increase in perplexity, which negatively impacts the modeling capabilities of the models to some extent. These outcomes resonate with the trade-off between accuracy and efficiency observed in Table 1.

A.2 Testing Acceleration Ratio to Full-Caching Method

In Section 5, we report the testing acceleration ratio following the setting of Wang et al. (2023a), comparing the time difference between caching and non-caching inference. Although our method reduces the keys/values caches, enabling smaller space for prefix information storage and improving testing time up to $\times 3.5$, we are still curious about whether it would enhance inference efficiency if conventional methods use full caches that save all keys/values of prefix tokens. As a supplement to Table 1, we present the testing acceleration ratio between anchor-caching and full-caching inference in Table 5. The acceleration ratios for AnLLM-EP-AnSAN and AnLLM-AC-AnSAN achieve the highest improvements observed in tasks such as HS, SCIQ, and BoolQ. The average acceleration ratios for AnLLM-EP-AnSAN and AnLLM-AC-AnSAN are 1.03. These results demonstrate that our anchor-based caching method can enhance inference efficiency even when compared to conventional methods that save all keys/values of prefix tokens. These results suggest that our anchor-based caching approach, which saves only the keys/values caches of anchor tokens, can effectively accelerate the inference process for the lengthy prefix texts.

A.3 Model Scalability Assessment

To examine the scalability of our approach, we extend the AnLLM-AC model to 13B and assess its performance on eight question-answering benchmarks using the same evaluation strategy as previously mentioned. In comparison to the 7B AnLLM models in Table 1, Results in Table 6 indicate that as the model size expands, the AnLLM-AC model achieves accuracies of 67.5% and 70.0% for 0-shot and 5-shot testing, respectively, resulting in up to a 2.4% improvement. Moreover, by incorporating anchor-based attention, the AnLLM-AC-AnSAN model achieves an average accuracy of 69.5%, signifying a 2.0% increase. The performance enhancement underscores the effectiveness of our methods in accommodating larger model capacities. The consistent improvements observed in the AnLLM-AC model across various scenarios highlight its robustness and adaptability. Furthermore, the increased performance of the AnLLM-AC-AnSAN model, facilitated by anchor-based attention, emphasizes the potential of our approaches in optimizing LLMs. Collectively, these findings point to promising avenues for future research aimed at maximizing the utility and efficiency of AnLLM.

A.4 Case Study in Real-Time Inference

To elaborate on the optimization of keys/values caches by AnLLM-EP and AnLLM-AC during real-time inference, we reference examples from the translation task in Section 6.2. As per Table 7, AnLLM-EP and AnLLM-AC use "endpoints" (".") and "<AC>" tokens as anchor tokens, respectively. During inference, both models employ auto-regressive generation, creating outputs token-by-token. Upon generating an anchor token (as per Line 16, Algorithm 1), the Reduction function (defined in Line 1) is activated, preserving relevant caches and eliminating others. As a result, the Keys/Values Cache lengths are reduced to roughly the sequence length, averaging around 50 for AnLLM-EP and 35 for AnLLM-AC, as shown in Table 2.

A.5 Attention Pattern in AnLLMs

Regarding the attention pattern, we have conducted a case study using the AnLLM-EP model. As shown in Figure 5, given the sentence "Apple is delicious. He goes to the market. He buys an apple.", we detokenize and split it into two segments: "_Apple _is _del icious . _He _go _to _the _market . _He _b" and "ys _an _apple .". By employing a heatmap to visualize the attention pattern between the latter segment and the former, we observe that the token "ys" attends more to the second endpoint, which compresses the information of "He goes to the market.". This is a reasonable and interesting

finding, as "ys" is part of the word "buys". Additionally, the token "_apple" attends more to the first endpoint, which compresses the information of "Apple is delicious.". These attention patterns offer some interpretability for our method.

B Data Settings

To provide a thorough insight into how we continually pre-train the model into AnLLM and carry out evaluations, we showcase some data examples in this section for both training and testing data.

B.1 Training Data Examples

In this section, we provide examples to illustrate the specific data format used in training the AnLLM models. For the AnLLM-EP model, the endpoints act as anchor tokens, allowing us to directly utilize natural language texts. For the AnLLM-AC model, we append a new token <AC> at the end of each sequence in the input texts, which are initially split into sentences using the NLTK toolkits.[5] Some examples are presented in Table 8. All the trainig data are downloaded from HuggingFace[6], an open-source community.

B.2 Testing Data Examples

For the testing outlined in the results section (Section 5), we employ the same evaluation method as in previous work (Gao et al., 2023), which treats each choice as text generation and computes the corresponding probabilities, respectively. Table 9 presents some evaluation examples.

[5] https://www.nltk.org/api/nltk.tokenize.punkt.html

[6] https://huggingface.co/datasets/togethercomputer/RedPajama-Data-1T-Sample

Method	Evaluation Context Length					
	256	**512**	**1024**	**2048**	**4096**	**AVG.**
Llama2-7B	5.42	4.32	3.64	3.2	2.91	3.90
AnLLM-AC	5.70	4.53	3.81	3.36	3.07	4.09
+AnSAN	6.61	5.61	5.01	4.62	4.40	5.25
AnLLM-EP	5.62	4.44	3.73	3.32	3.04	4.04
+AnSAN	6.18	5.17	4.62	4.31	4.14	4.88

Table 4: Perplixity Evaluation without or with the AnSAN Technique. The test samples are from the Proof-Pile dataset.

	OBQA	**WG**	**ARC-e**	**ARC-c**	**PIQA**	**HS**	**SCIQ**	**BoolQ**	**AVG.**
AnLLM-EP-AnSAN	×1.00	×1.00	×1.00	×1.00	×1.00	×1.06	×1.14	×1.13	×1.03
AnLLM-AC-AnSAN	×1.00	×1.02	×1.00	×1.00	×1.00	×1.01	×1.10	×1.13	×1.03

Table 5: Testing Acceleration Ratio on Question-Answering Tasks between Anchor-Caching and Full-Caching Inference with Five-Shot Demonstrations. Anchor-caching refers to saving only the keys/values caches of anchor tokens with the AnSAN technique, while full-caching denotes saving caches for all prefix tokens. The tasks are arranged according to the demonstration lengths. The experiments are the same as those of Table 1. These results suggest that inference speed differences for short texts are minimal but become more pronounced for longer texts. However, full-caching inference demands more GPU memory to store the complete keys/values caches, which is not ideal for environments with limited computational resources.

Model	**OBQA**	**WG**	**ARC-e**	**ARC-c**	**PIQA**	**HS**	**SCIQ**	**BoolQ**	**AVG.**
Zero-Shot Performance									
Llama2-7B	31.4	69.1	76.3	43.4	78.1	57.1	93.7	77.7	65.8
Llama2-13b	35.2	72.1	79.4	48.5	79.1	60.0	94.5	80.6	68.7
AnLLM-AC-7B	31.6	68.5	74.4	42.5	78.3	54.7	93.8	77.0	65.1
AnLLM-AC-13B	35.2	70.7	77.9	46.9	78.6	58.1	94.7	78.1	67.5
Five-Shot Performance									
Llama2-7B	37.2	73.7	79.8	50.0	78.7	58.3	96.8	78.4	69.1
Llama2-13b	38.2	76.3	82.2	52.6	80.0	61.4	97.5	83.5	71.5
AnLLM-AC-7B	37.2	72.3	79.8	49.0	78.6	56.9	96.8	77.5	68.5
+AnSAN	35.6	70.6	79.2	47.9	78.7	55.6	95.7	76.6	67.5
AnLLM-AC-13B	36.6	72.5	81.6	53.7	79.2	59.6	97.5	79.6	70.0
+AnSAN	36.0	74.0	81.6	52.0	79.1	58.4	96.3	78.8	69.5

Table 6: Accuracy of 13B LLMs on Question Answering Benchmarks. Compared to 7B AnLLMs, the 13B AnLLMs exhibit superior performance, with up to 2.0 accuracy enhancements, suggesting that AnLLMs possess excellent scalability to larger model architectures.

Input	Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. ### Instruction: Translate the following sentences from German to English. ### Input: Nachdem Werte in einen anderen Teil des Speichers eingeschrieben wurden, wird das CMOS RAM in der gleichen Weise wie das Communications RAM geprüft. Wurde der Test bestanden, werden alle Speicherstellen auf ihren früheren Wert eingestellt. LED-Wert: 00 0011 Wenn das Gerät mit dem o.a. Display hält, liegt ein Fehler vor. Prüfen Sie in diesem Falle U 85 und U 86 und die damit verbundenen Stromkreise bzw. die Dekodierung. ### Response:
Output	After values have been written to another part of the CMOS RAM, the CMOS RAM is tested in the same way as the communications RAM. If the test is successful, all storage locations will be reset to their former value. LED value: 00 0011 If the device is displaying this value, there is a fault to be found. In this case, check U85 and U86 and the associated power circuits, as well as decoding.
COMET-DA	82.2
Length	293
MaxKV	170
$\mathcal{C}_{\Downarrow}$	42%

(a) An Example of the AnLLM-EP Model in De2En Translation Task.

Input	Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. <AC> ### Instruction: Translate the following sentences from German to English. <AC> ### Input: Der Spielablauf wird ausgelöst, wenn der Druckschalter START betätigt wird. <AC> Der LED-Schalter bleibt erleuchtet und die Spielgeräusche werden hörbar. <AC> Wird der letzte Spieler-Panzer vom feindlichen Angreifer eliminiert, ist ein Spiel beendet. <AC> Das Spiel beginnt mit langsam fahrenden Panzern, die auf dem Bildschirm sichtbar werden. <AC> Die Position der jeweiligen feindlichen Panzer oder Raketen kann der Spieler dadurch bestimmen, dass er den eingeblendeten Text im oberen linken Hinweisdisplay-Winkel seines Cockpits beachtet. <AC> Ein Hinweis kann z.B. <AC> lauten: GEGNER VON RECHTS. <AC> Zusätzlich wird auf dem weitreichenden Radarschirm, begleitet von einem Sonar-BLIP-Ton, ein Punkt sichtbar, der die feindliche Stellung darstellt. <AC> ### Response:
Output	The game is triggered when the start switch is pressed. <AC> The LED switch remains illuminated and the game sounds can be heard. <AC> When the last enemy player's tank is destroyed, the game is over. <AC> The game begins with tanks moving slowly move across the screen. <AC> The player's tank's position can be determined by the player by paying attention to the respective enemy tank or missile icon in the upper left corner of his cockpit's display screen. <AC> An example of an instruction can be: ENEMY ON THE RIGHT. <AC> In addition, on the wide-range radar screen, accompanied by a sonar BLIP sound, a point is visible, which indicates the enemy position.
COMET-DA	81.8
Length	431
MaxKV	272
$\mathcal{C}_{\Downarrow}$	37%

(b) An Example of the AnLLM-AC Model in De2En Translatio Task.

Table 7: Case Study of Real-time Inference. During the inference process, AnLLM-EP generates "endpoint" as the anchor token, whereas AnLLM-AC produces "<AC>" as the anchor token. Once upon an anchor token, we execute the REDUCTION as shown in Line 16 to reduce the keys/values caches.

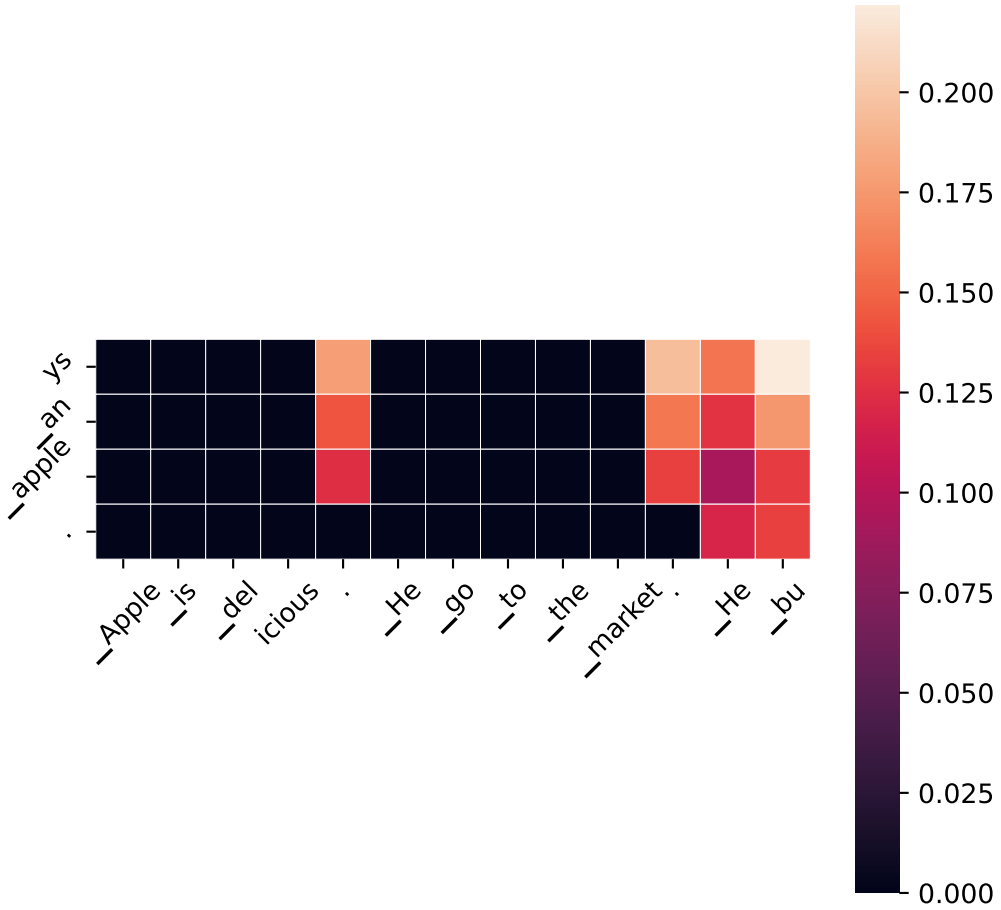

Figure 5: Case Study on Attention Pattern. With the AnLLM-EP model, the heatmap presents the average correlation between tokens across layers of output hidden states.

Gender diversity, or more often its lack thereof, among participants to software development activities has been thoroughly studied in recent years. In particular, the presence of, effects of, and countermeasures for gender bias in Free/Open Source Software (FOSS) have received a lot of attention over the past decade. Geographic diversity is on the other hand the kind of diversity that stems from participants in some global activity coming from different world regions and cultures. Geographic diversity in FOSS has received relatively little attention in scholarly works. In particular, while seminal survey-based and point-in-time medium-scale studies of the geographic origins of FOSS contributors exist, large-scale longitudinal studies of the geographic origin of FOSS contributors are still lacking. Such a quantitative characterization would be useful to inform decisions related to global development teams and hiring strategies in the information technology (IT) market, as well as contribute factual information to the debates on the economic impact and sociology of FOSS around the world. ...

(a) A Training Data Example for the AnLLM-EP Model. The endpoints in the text serve as the anchor tokens.

Gender diversity, or more often its lack thereof, among participants to software development activities has been thoroughly studied in recent years. <AC> In particular, the presence of, effects of, and countermeasures for gender bias in Free/Open Source Software (FOSS) have received a lot of attention over the past decade. <AC> Geographic diversity is on the other hand the kind of diversity that stems from participants in some global activity coming from different world regions and cultures. <AC> Geographic diversity in FOSS has received relatively little attention in scholarly works. <AC> In particular, while seminal survey-based and point-in-time medium-scale studies of the geographic origins of FOSS contributors exist, large-scale longitudinal studies of the geographic origin of FOSS contributors are still lacking. <AC> Such a quantitative characterization would be useful to inform decisions related to global development teams and hiring strategies in the information technology (IT) market, as well as contribute factual information to the debates on the economic impact and sociology of FOSS around the world. <AC> ...

(b) A Training Data Example for the AnLLM-AC Model. The newly added tokens <AC> in the text serve as the anchor tokens.

Table 8: Training Data Examples for the AnLLM-EP and AnLLM-AC models. For the AnLLM-EP model, the endpoints are the natural anchor tokens. For the AnLLM-AC model, we manually append <AC> tokens to sequences as the anchor tokens.

Choice 1: Slacklining: A group of people have stretched a tightrope across a gym. They *take turns trying to balance and walk on the rope.*
Choice 2: Slacklining: A group of people have stretched a tightrope across a gym. They *slide down with it, jumping and spinning in the air.*
Choice 3: Slacklining: A group of people have stretched a tightrope across a gym. They *cross it together, swinging back and fourth in anticipation.*
Choice 4: Slacklining: A group of people have stretched a tightrope across a gym. They *drop an orange rope at the end.*

(a) A Zero-Shot Testing Data Example of the HellaSwag Task. The log-likelihood of the red texts is computed as the choice probabilities.

Choice 1: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. Demonstration 2 Demonstration 3 Demonstration 4 Demonstration 5 Slacklining: A group of people have stretched a tightrope across a gym. They *take turns trying to balance and walk on the rope.*
Choice 2: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. Demonstration 2 Demonstration 3 Demonstration 4 Demonstration 5 Slacklining: A group of people have stretched a tightrope across a gym. They *slide down with it, jumping and spinning in the air.*
Choice 3: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. Demonstration 2 Demonstration 3 Demonstration 4 Demonstration 5 Slacklining: A group of people have stretched a tightrope across a gym. They *cross it together, swinging back and fourth in anticipation.*
Choice 4: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. Demonstration 2 Demonstration 3 Demonstration 4 Demonstration 5 Slacklining: A group of people have stretched a tightrope across a gym. They *drop an orange rope at the end.*

(b) A Five-Shot Testing Data Example of the HellaSwag Task for the ALLM-EP Inference.

Choice 1: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. <AC> Demonstration 2 <AC> Demonstration 3 <AC> Demonstration 4 <AC> Demonstration 5 <AC> Slacklining: A group of people have stretched a tightrope across a gym. They *take turns trying to balance and walk on the rope.*
Choice 2: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. <AC> Demonstration 2 <AC> Demonstration 3 <AC> Demonstration 4 <AC> Demonstration 5 <AC> Slacklining: A group of people have stretched a tightrope across a gym. They *slide down with it, jumping and spinning in the air.*
Choice 3: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. <AC> Demonstration 2 <AC> Demonstration 3 <AC> Demonstration 4 <AC> Demonstration 5 <AC> Slacklining: A group of people have stretched a tightrope across a gym. They *cross it together, swinging back and fourth in anticipation.*
Choice 4: Ballet: We see a pregnant lady doing ballet in a studio. The lady spins and does a pliea. <AC> Demonstration 2 <AC> Demonstration 3 <AC> Demonstration 4 <AC> Demonstration 5 <AC> Slacklining: A group of people have stretched a tightrope across a gym. They *drop an orange rope at the end.*

(c) A Five-Shot Testing Data Example of the HellaSwag Task for the ALLM-AC Inference.

Table 9: Testing Data Examples for the AnLLM-EP and AnLLM-AC models. The log-likelihood of the red italicized texts is calculated as the choice probabilities.