FocusLLM: Precise Understanding of Long Context by Dynamic Condensing

Anonymous ACL submission

Abstract

Empowering LLMs with the ability to precisely understand long contexts is crucial for many downstream applications. However, handling long contexts with conventional transformer architecture requires substantial training and inference resources. Existing context condensing methods cannot accurately understand the full context, as there is a considerable amount of information loss in the condensing process. To address these issues, we present **FocusLLM**, a framework designed to extend the fixed context length of any decoder-only LLM, allowing the model to focus on relevant information from very long sequences. FocusLLM first divides long text input into chunks based on the model's original context length. It then employs the ***dynamic condensing*** process to distill crucial information from each chunk. Ultimately, through the novel ***parallel decoding*** mechanism, FocusLLM can integrate the extracted information into its local context. FocusLLM stands out for great training efficiency and versatility: trained with an 8K input length and with much less training cost than previous methods, FocusLLM exhibits superior performance across downstream tasks and maintains strong language modeling ability when handling extensive long texts, even up to 400K tokens. Our code is available at `https://anonymous.4open.science/r/FocusLLM`.

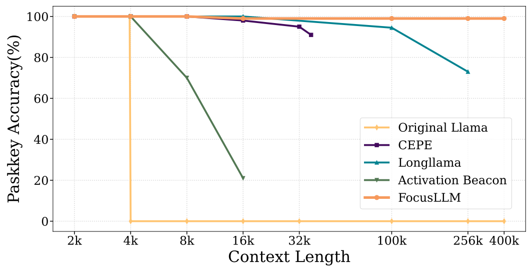

Figure 1: A comparison between FocusLLM and previous context scaling methods on the passkey retrieval task, including CEPE, LongLLaMA and Activation Beacon. Our method extrapolates beyond the original context length of LLaMA, achieving 99% accuracy at a context length of 400K, with less training cost.

1 Introduction

The importance of extending the context length of large language models (LLMs) cannot be overstated. In numerous applications, ranging from complex document analysis to generating coherent long-form text, the ability to effectively utilize extended context is critical. For instance, in tasks such as document summarization and question answering over lengthy articles, a more extensive context allows for a more comprehensive understanding and accurate responses. However, leveraging long contexts in LLMs presents several formidable challenges. (1) The computational complexity of transformers (Vaswani et al., 2017) grows quadratically with the sequence length, rendering the training process prohibitively expensive. (2) LLMs exhibit poor extrapolation performance for longer sequences, even after additional fine-tuning (Chen et al., 2023a; Peng et al., 2023). (3) Acquiring high-quality long-text datasets, which are essential for training and fine-tuning, is exceedingly difficult (Xiong et al., 2023; Wang et al., 2022).

To circumvent the substantial costs of directly scaling the window length by continual training on longer inputs, recent work has proposed to drop unimportant tokens and retain important tokens, either by modifying the attention mechanism (Xiao et al., 2023; Han et al., 2023) or by compressing the context into some specialized tokens (Zhang et al., 2024a; Chevalier et al., 2023; Ge et al., 2023), in order to effectively condense long textual information. However, these methods overlook the fact that *token importance changes dynamically during the decoding process*: tokens previously considered unimportant may become crucial in later decoding steps. As a result, they share a common drawback, which we refer to as ***information loss***: some tokens that will be needed in the future have already been discarded. For example, in Passkey Retrieval task (Mohtashami and Jaggi, 2024) illustrated in Figure 1, as the context length increases, the compression method Activation Beacon fails to retrieve passkey pairs that appeared in the earlier context.

Considering the above issues, the question arises: *can we extend the context length of an existing LLM at a low cost without any information loss?* In this paper, we propose a training efficient and effective solution ***FocusLLM***, which can maintain a precise understanding of the whole long context. Specifically, FocusLLM first divides a long text into chunks based on the model's original context length. Then, the ***dynamic condensing*** process is applied, which appends dynamic prompts to each chunk to extract crucial information, ensuring no information loss. Finally, we use ***parallel decoding*** mechanism to aggregate information from different chunks and generate the next token. The original model parameters are kept frozen to maintain generalization capabilities, with only a small number of trainable parameters introduced for dynamic condensing.

We employ the FocusLLM framework to the widely used LLaMA-2-7B model (Touvron et al., 2023b), which has a default context length of 4K. In terms of efficiency, FocusLLM is trained on sequences shorter than 8K tokens and only requires a training budget of ***0.5B tokens***. To validate the effectiveness of FocusLLM, we evaluate it across a variety of tasks. Initially, we assessed FocusLLM's language modeling capability. FocusLLM maintains low perplexity on documents comprising 128K tokens and even longer sequences. Subsequently, to comprehensively evaluate the applicability of FocusLLM in real-world scenarios, we utilized two widely used benchmarks: Longbench (Bai et al., 2023) and ∞-Bench (Zhang et al., 2024b). Experimental results demonstrate that FocusLLM has achieved superior performance on both benchmarks, surpassing all baselines including length extrapolation models, continual training models, and similar models designed for extreme long sequences. The main contributions of this paper can be summarized as follows:

- We propose the FocusLLM framework, which leverages novel ***dynamic condensing*** and ***parallel decoding*** mechanisms to avoid information loss and achieve precise understanding of long contexts, as shown in Figure 1.
- Compared to previous context-scaling methods, FocusLLM achieves remarkable results with ***high training efficiency*** by introducing only a small set of trainable parameters and utilizing a training budget of 0.5B tokens.
- Through comprehensive evaluation, FocusLLM outperforms all baselines on downstream tasks while maintaining low perplexity, demonstrating that it can seamlessly serve as a general-purpose language model.

2 Architecture

The overall framework of FocusLLM is presented in Figure 2. Each decoder in the figure shares the same model (e.g. LLaMA-2).

2.1 Notations

Given a long sequence with S tokens $\{x_1, ..., x_S\}$, we segment them into **memory tokens** $\{x_1, ..., x_m\}$ and **local context** $\{x_{m+1}, ..., x_S\}$, with the length of local context not exceeding the model's default context length, denoted as L. Concurrently, we divide the memory tokens into **chunks**, labeled as $C_1, C_2, ..., C_k$, with each chunk's size also not exceeding L. These chunks can represent distinct documents or a single long document. We define the original decoder model as F_{dec} and its hidden dimension d_{dec}. To endow the model with the capability for dynamic condensing, we introduce a small set of new parameters, resulting in the modified model F'_{dec}.

2.2 Dynamic Condensing

As highlighted in the introduction, the importance of tokens in the context dynamically changes at each decoding step. Previous work that condenses context using a fixed pattern suffers from the drawback of *information loss*. To address this issue, we propose the dynamic condensing mechanism, which consists of two key steps: dynamic prompt injection and candidate token generation.

Dynamic Prompt Injection. We append a small fragment of local context (we refer to it as the *dynamic prompt* in Figure 2) behind each chunk. The motivation is to aggregate the most critical information from each chunk for the current decoding step. We can formally define this process as follows:

$$\hat{C}_i \leftarrow \{C_i; x_{m+j}, ..., x_S\} \; i = 1, ..., k; 1 \leq j \leq S - m \quad (1)$$

Here j is a hyperparameter that determines the number of local tokens appended to each chunk. We

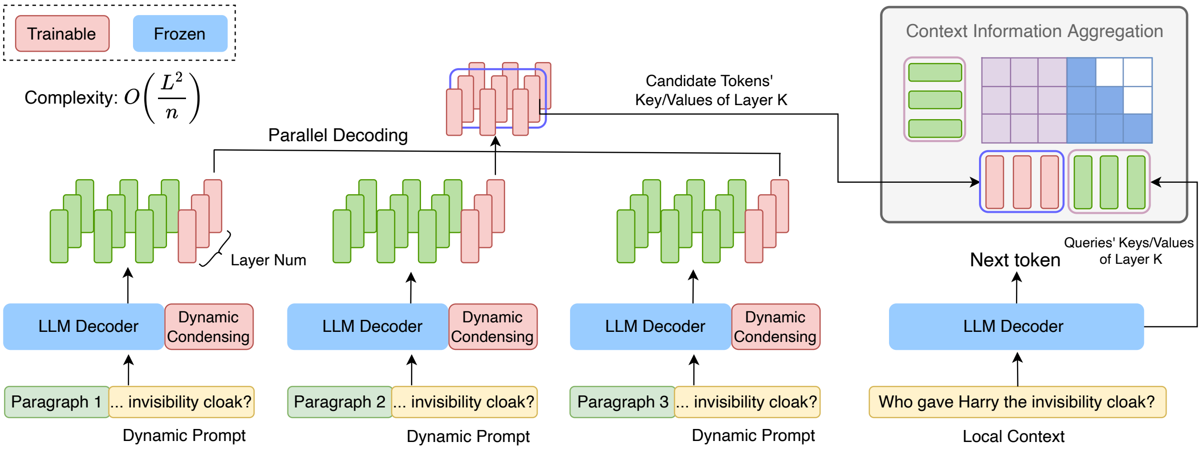

Figure 2: One decoding step of the FocusLLM framework. A small fragment of the local context (denoted as the dynamic prompt) is appended to each chunk. The representations of the candidate tokens, obtained through dynamic condensing and parallel decoding, are then concatenated and integrated back into the local context.

adopt a default length of 512 tokens for inference, which is sufficient to encapsulate the necessary local contextual information.

The last token of the dynamic prompt is used to generate candidate tokens, which we will explain in detail later. After each decoding step, when FocusLLM generates the next token, this token will be appended to the dynamic prompt [1]. This updated dynamic prompt is then used to generate new candidate tokens in the next decoding step.

The *dynamic prompt* evolves with each decoding step, ensuring that the model always has access to the most relevant information for the current step.

Candidate Token Generation. Building on the dynamic prompt injection described above, we introduce *candidate tokens* to condense the information from each chunk that is crucial for the current decoding step. The *candidate token* is denoted as the trainable hidden states corresponding to the last local token x_S in each chunk $\hat{C}_i$. To obtain the representations of candidate tokens, motivated by (Zhang et al., 2024a), we add a new set of trainable parameters to the linear projection matrices of each layer, while keeping the original model parameters frozen to preserve its original decoding ability. Formally, the trainable parameters for dynamic condensing are:

$$\{W_Q^c, W_K^c, W_V^c, W_O^c\}_l \tag{2}$$

where W_Q^c, W_K^c, W_V^c, and W_O^c represent the new linear projections for the query, key, value, and output matrices associated with the candidate token, and l denotes the layer number. The output of the candidate token in the self-attention module can be calculated as:

$$Q_c \leftarrow H_c W_Q^c \quad K_c \leftarrow H_c W_K^c \quad V_c \leftarrow H_c W_V^c \tag{3}$$

$$A_c \leftarrow \mathrm{softmax}\left(Q_c \left(K \oplus K_c\right)^T\right) \tag{4}$$

$$O_c \leftarrow V_c {W_O^c}^T \quad V_c \leftarrow A_c \left(V \oplus V_c\right)^T \tag{5}$$

where $H_c \in \mathbb{R}^{d_{dec}}$ is the input hidden state of the candidate token, $\oplus$ represents the concatenation of matrices, and K, V correspond to the representations of the normal tokens in one chunk.

2.3 Parallel Decoding

Through the dynamic condensing process described above, we obtain one candidate token for each chunk. Notably, the process of obtaining the candidate token from each chunk is independent, enabling *parallel forwarding* for all chunks. Then the key/value representations of the candidate tokens are concatenated with the tokens in the local context layer by layer, as shown in Figure 2, and are finally processed by a frozen decoder to generate the next token.

We formally define the process of simultaneously generating candidate tokens from different chunks and then aggregating these candidate tokens to produce the final token as *parallel decoding*. This mechanism not only enables precise understanding of long contexts but also reduces the Transformer's original $O(L^2)$ computational complexity to $O((L/n)^2)$. A detailed efficiency analysis is provided in Appendix A.

[1]The first token of the dynamic prompt can be dropped to maintain its fixed length.

3 Training

Regarding training data, to ensure the generalizability of our method and maintain fairness in comparison with the baselines, we leverage RedPajama (Together, 2023b) as the training corpus and sample examples with sequence lengths varying between 3K and 8K tokens from it. RedPajama is an open-source pre-training dataset for LLaMA-1 (Touvron et al., 2023a), which is widely utilized in previous work (Zhang et al., 2024a; Yen et al., 2024). Detailed statistics are reported in Appendix B.

Auto-Regressive Loss. Specifically, we train the model to predict the next token, and the loss is only applied to tokens in the local context, which encourages the candidate token to aggregate useful information from each chunk.

$$\min_{F'_{\text{dec}}} - \sum_{i=2}^{S-m} \log(p(x_{m+i} \mid c_1, \ldots, c_k, x_{m+1}, \ldots, x_{m+i-1})) \tag{6}$$

Here, c_i represents the candidate token generated by the i-th chunk. Specifically, based on the relationship between the *memory tokens* $\{x_1, ..., x_m\}$ and the *local context* $\{x_{m+1}, ..., x_S\}$, we design two loss functions for joint training. **i)** If the local context is a continuation of the memory tokens, we term this loss the *Continuation Loss*, as it trains the model to naturally generate new tokens that follow the given context. **ii)** Alternatively, if we randomly select L consecutive memory tokens as local context, we define this loss as the *Reconstruction Loss*, as it trains the model to reconstruct tokens when clear contextual information is available. Subsequent experiments demonstrate that both types of loss are essential.

4 Experiments

In this section, we will conduct a comprehensive evaluation of the effectiveness of FocusLLM, spanning both language modeling and a variety of downstream tasks. We refer readers to Appendix C for detailed experimental settings including hyperparameters due to space constraints.

4.1 Long-context Language Modeling

In this section, we evaluate FocusLLM on long-context language modeling benchmarks, with text lengths ranging from 4K to 128K tokens.

Datasets. We perform the evaluation on three datasets: PG19 (Rae et al., 2019), Proof-Pile (Azerbayev et al., 2023), and CodeParrot (Tunstall et al., 2022). These three datasets encompass 100 long test cases related to books, arXiv papers, and code repositories, respectively. The results of baseline models are token from (Zhang et al., 2024a) for comparison. Following the setting of (Yen et al., 2024), as FocusLLM relies on the last decoder to perform generation, we calculate the perplexity on the last 256 tokens of each sequence, and for the 128K length, we filter out documents exceeding 128K tokens and evaluate 10 samples due to data scarcity and computational cost.

Model. FocusLLM is based on LLaMA-2-7B (chat), hence the models for comparison are all on the same scale, 7B. The baseline models can be categorized into the following types: i) Methods focusing on the modification of **positional encoding**, including Positional Interpolation (Chen et al., 2023a), the NTK-Aware Scale ROPE[2], and the training-free method StreamingLLM (Xiao et al., 2023), which is based on attention sinks. ii) **Fine-tuned methods** trained on long inputs, such as LongAlpaca-16K (Chen et al., 2023b), LongChat-32K (Li et al., 2023), and YaRN-128K (Peng et al., 2023). iii) **Methods with designed structures** specifically for long contexts, including AutoCompressor-6K (Chevalier et al., 2023), LongLlama (Tworkowski et al., 2024) and Activation Beacon (Zhang et al., 2024a). For instance, Activation Beacon achieves compression of long texts by training the model to represent the information of a regular text segment with a small number of beacon tokens.

Analysis. The results are presented in Table 1. Here are several observations we can make: (1) Compared to the basic LLaMA-2-7B model and some fine-tuning free methods, our model demonstrates superior performance. When extending the context length from 4K to longer, the perplexity becomes lower, indicating that information from a longer context can be effectively utilized. (2) FocusLLM achieves comparable performance to fine-tuned full-attention methods. This result is notable because our model operates with significantly higher training efficiency. For instance, LongLlama is fine-tuned using 7B tokens with all parameters being trainable. In contrast, FocusLLM uses $1/10$ of the training budget and $1/3$ of the parameters. (3) FocusLLM can maintain language modeling capabilities at lengths much longer than other models while retaining precise comprehension of the

[2] https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/

Method	PG19 4K	PG19 16K	PG19 32K	PG19 100K	Proof-Pile 4K	Proof-Pile 16K	Proof-Pile 32K	Proof-Pile 100K	CodeParrot 4K	CodeParrot 16K	CodeParrot 32K	CodeParrot 100K
Llama-2-7B	9.21	$>10^3$	$>10^3$	OOM	3.47	$>10^3$	$>10^3$	OOM	2.55	$>10^3$	$>10^3$	OOM
PI	9.21	19.5	$>10^2$	OOM	3.47	5.94	33.7	OOM	2.55	4.57	29.33	OOM
NTK	9.21	11.5	37.8	OOM	3.47	3.65	7.67	OOM	2.55	2.86	7.68	OOM
StreamingLLM	9.21	9.25	9.24	9.32	3.47	3.51	3.50	3.55	2.55	2.60	2.54	2.56
AutoCompre.-6K	11.8	$>10^2$	$>10^3$	OOM	4.55	$>10^2$	$>10^3$	OOM	5.43	$>10^2$	$>10^3$	OOM
YaRN-128K	6.68	6.44	6.38	OOM	2.70	2.47	2.41	OOM	2.17	2.04	2.00	OOM
LongChat-32K	9.47	8.85	8.81	OOM	3.07	2.70	2.65	OOM	2.36	2.16	2.13	OOM
LongAlpaca-16K	9.96	9.83	$>10^2$	OOM	3.82	3.37	$>10^3$	OOM	2.81	2.54	$>10^3$	OOM
LongLlama	9.06	8.83	OOM	OOM	2.61	2.41	OOM	OOM	1.95	1.90	OOM	OOM
Activation Beacon	9.21	8.54	8.56	8.68	3.47	3.42	3.39	3.35	2.55	2.54	2.53	2.55
FocusLLM	9.21	9.19	9.17	10.59	3.47	3.17	3.43	2.57	2.55	2.01	2.27	3.02

Table 1: Language Modeling Assessment: perplexity analysis of various context scaling methods on the PG19, Proof-Pile, and CodeParrot. FocusLLM successfully maintains low perplexity on extremely long sequences.

entire text. Although models like StreamingLLM and Activation Beacon can still achieve lower perplexity by compressing tokens, they are unable to recover the previous context information, which severely affects their capabilities in downstream tasks. In summary, FocusLLM achieves comparable language modeling performance with a small training cost.

4.2 Downstream Tasks

Datasets. To assess the capabilities of FocusLLM in real-world scenarios, we select two widely used datasets: Longbench (Bai et al., 2023) and ∞-Bench (Zhang et al., 2024b). Longbench offers an evaluation on a variety of tasks including question answering, summarization, few-shot learning, mathematical counting, and code completion. ∞-Bench is designed to test a model's ability to understand and reason over super long contexts, with an average length of 145.1K tokens. Thus, the tasks in ∞-Bench are well-suited to test whether the model has a precise understanding of long contexts without ***information loss***. For more detailed statistics, please refer to Appendix D. We believe that these two benchmarks can comprehensively reflect the capabilities of the model on downstream tasks.

Models. We select representative models from the three types of baselines mentioned in Section 4.1 for comparison. Additionally, we focus on comparing FocusLLM with recently proposed models capable of processing extremely long streaming inputs. Specifically, StreamingLLM utilizes a sliding window mechanism; InfLLM (Xiao et al., 2024) stores processed context into memory units and retrieves it using attention scores; Activation Beacon compresses the preceding text to maintain a smaller context length. CEPE (Yen et al., 2024) adopts a small encoder to process long inputs chunk by chunk and feeds the memory to a decoder by cross-attention.

Main Results. The experimental results are displayed in Table 2 and 3. We reference some baseline results from (Xiao et al., 2024), which are based on the Vicuna-7B-v1.5 model. Vicuna-7B-v1.5 is based on LLaMA-2-7B but fine-tuned on conversational data. For a fair comparison, we also train a Vicuna version of FocusLLM. For YaRN-128K, we select the version based on Mistral-7B-inst-v0.2, which is stronger than Vicuna. For LongLlama, as they do not have a version based on the Llama2, we directly utilize the officially released model. CEPE and LongLLaMA will experience *OOM* on ∞-Bench due to their substantial memory usage, so we only report their results on LongBench. Since not all models are inherently capable of processing infinite text lengths, we also elaborate the effective lengths for each method presented in Tables 2 and 3 in Appendix E.

From the experimental results, we can make the following comparisons between FocusLLM and previous methods: **(1)** FocusLLM outperforms all baseline models, achieving ***the best results*** on both the relatively shorter benchmark Longbench and the extremely long benchmark ∞-Bench. This demonstrates FocusLLM's capability for effective understanding and reasoning on long sequences and its broad applicability. **(2)** Different types of baseline models exhibit various shortcomings. For training-free models like PI and NTK, extending the length to 128K comes with a significant sacrifice in performance. Due to the lack of precise understanding of the full context, models that

		Vicuna-7B-v1.5 (4K)								
		Original	LChat	Vic-16K	Yarn-128K	PI	NTK	Stream	InfLLM	FocusLLM
∞-Bench	Math.Find	11.71	9.43	13.43	17.14	OOM	OOM	6.00	11.14	**11.71**
	En.MC	30.13	24.45	34.06	27.95	OOM	OOM	**32.31**	31.44	**32.31**
	Code.Debug	38.83	27.66	35.03	22.59	OOM	OOM	**46.19**	34.26	28.43
	Retrieve.KV	1.40	1.40	1.00	0.00	OOM	OOM	0.00	0.60	**12.40**
	Retrieve.Number	4.41	23.90	10.34	56.61	OOM	OOM	4.41	81.69	**83.56**
	Retrieve.PassKey	5.08	28.64	15.25	92.71	OOM	OOM	4.92	**99.15**	95.76
	Average	15.26	19.25	18.19	36.17	–	–	15.64	43.05	**44.03**
LongBench	NarrativeQA	11.19	20.35	17.85	19.67	0.78	5.66	15.61	15.53	**21.14**
	Qasper	13.79	29.35	25.85	11.10	2.71	21.17	23.84	23.57	**31.07**
	MultiFieldQA	22.08	42.55	37.15	35.06	1.01	36.76	32.80	**37.14**	36.73
	HotpotQA	12.71	33.19	24.72	11.94	1.35	19.54	22.17	22.53	**40.65**
	2WikiMQA	13.99	24.33	21.41	12.02	1.17	14.51	18.38	18.82	**20.30**
	Musique	4.81	14.71	8.44	7.52	0.71	4.30	6.30	5.24	**14.20**
	GovReport	27.67	30.83	27.62	29.46	1.9	25.26	23.18	**26.79**	26.66
	QMSum	19.72	22.93	22.63	21.53	1.29	19.48	20.09	**20.91**	20.50
	MultiNews	26.61	26.63	27.88	16.04	1.16	25.88	26.19	26.43	**27.45**
	TREC	69.00	66.50	69.00	68.50	4.50	59.00	61.00	**67.50**	**68.00**
	TriviaQA	81.94	83.99	85.63	88.21	0.90	25.85	78.81	**84.36**	81.63
	SAMSum	35.12	12.83	9.15	26.52	0.12	5.05	32.46	31.89	**35.36**
	PassageRetrieval	9.00	30.50	4.00	16.25	0.62	5.00	6.00	9.00	**15.67**
	LCC	64.53	54.79	50.64	66.39	21.54	53.65	**63.70**	61.41	62.79
	RepoBench-P	50.17	58.99	44.94	55.82	19.36	44.58	48.26	47.52	**53.72**
	Average	30.82	34.70	31.79	32.40	3.94	24.38	31.92	33.24	**36.17**

Table 2: The results on ∞-Bench and LongBench. The models on the right part can process extremely long inputs. On both benchmarks, FocusLLM achieves significant improvements compared to strong baselines.

employ sliding window or condensing techniques, such as StreamingLLM and Activation Beacon perform poorly on ∞-Bench (see also Appendix F), with performance nearly approaching zero on some tasks. This indicates that ***they suffer from severe information loss***. As for fine-tuned models like LongChat and CEPE, their limitation is the restricted supported length. For example, CEPE struggles to handle lengths beyond 128K effectively (Yen et al., 2024). **(3)** The approaches of length extrapolation and continual training on long inputs, while capable of scaling context, introduce substantial computational and memory costs. In contrast, FocusLLM processes the text in chunks and utilizes parallel decoding, which significantly conserves both the memory and time for inference.

5 Further Exploration

5.1 Visualization of Candidate Tokens

To further illustrate how candidate tokens function, we provide a more intuitive explanation by visualizing the information carried by these tokens through attention weight heatmaps when decoding the next token. Due to space limitations, we place the visualization results in Appendix H. We have the following observations: i) In Passkey Retrieval task, the model assigns a high attention weight to *one certain candidate token*, indicating that this token effectively carry the passkey information from its respective chunk. In contrast, candidate tokens from chunks containing noisy text carry no useful information, resulting in near-zero attention weights. ii) In LongBench NarrativeQA task, the model shows a slightly different pattern, where *many candidate tokens receive attention*, as multiple chunks' information may be aggregated for the QA task. The visualization results demonstrate that FocusLLM effectively uses candidate tokens to transmit information from the context while ignoring irrelevant noise.

5.2 Scaling to 400K Context

We contend that FocusLLM is capable of processing extremely long sequences. To validate this, we first conduct experiments on the passkey retrieval task (Mohtashami and Jaggi, 2024). The results, as illustrated in Figure 1, demonstrate that FocusLLM maintains nearly 100% effectiveness at lengths of up to 400K[3], outperforming all other models. We also extended the language modeling experiments introduced in Section 4.1 to 400K, a length at which most models fail to manage effectively. The result is presented in the Appendix G.

[3]Constrained by hardware, the maximum length we are able to test is 400k tokens.

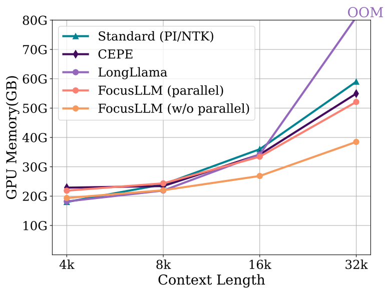

Figure 3: FocusLLM exhibits a more efficient growth pattern in memory usage compared to previous methods.

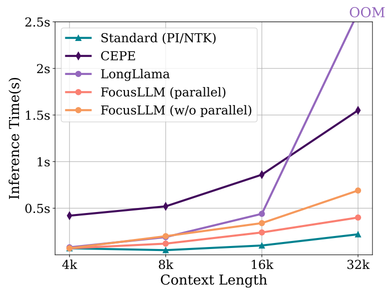

Figure 4: Comparison of inference time. The time taken by FocusLLM is superior to previous methods.

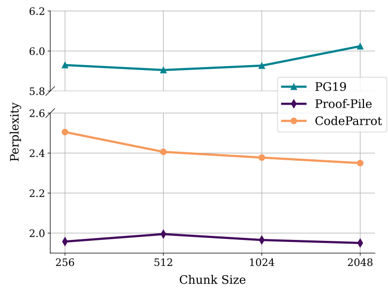

Figure 5: Perplexity under different chunk size with the total sequence length fixed as 8K on three datasets.

	Llama-7B-chat (4K)				
	Original	CEPE	L_L	A_B	FocusLLM
NarrativeQA	18.70	22.14	-	-	20.38
Qasper	19.20	26.34	-	-	21.73
MultiFieldQA	36.80	31.56	-	-	36.91
-Average	24.90	26.68	**30.12**	27.14	26.34
HotpotQA	25.40	34.95	-	-	38.95
2WikiMQA	32.80	32.39	-	-	32.95
Musique	9.40	9.76	-	-	15.39
-Average	22.60	25.70	16.37	28.28	**29.10**
GovReport	27.30	13.90	-	-	25.54
QMSum	20.80	20.30	-	-	21.86
MultiNews	25.80	3.10	-	-	26.35
-Average	24.70	12.43	24.19	**25.15**	24.55
TREC	61.50	68.50	-	-	68.00
TriviaQA	77.80	87.90	-	-	85.08
SAMSum	40.70	32.38	-	-	41.63
-Average	60.00	62.92	60.31	60.72	**64.81**
LCC	52.40	66.21	-	-	58.42
RepoBench-P	43.80	58.94	-	-	54.27
-Average	48.10	62.57	**66.05**	57.83	56.35
Average	35.20	36.31	37.50	38.54	**39.01**

Table 3: The results of LLaMA2-based models on tasks of LongBench. L_L represents Long Llama and A_B represents Activation Beacon. FocusLLM outperforms memory-based and compression-based methods, and maintains attention to all tokens of context.

5.3 Memory Footprint and Inference Time

For models that focus on long texts, aside from training costs, another critical aspect is the memory footprint and inference time. In this section, we compare FocusLLM with several previous long-context methods capable of *retaining global information by preserving the cache* of all context: Standard (PI/NTK), LongLlama, and CEPE. As for models like Activation Beacon and StreamingLLM, although they maintain a constant memory footprint by only retaining cache for a fixed window, they suffer significant *information loss* and struggle with the precise understanding of long texts as demonstrated in Section 4.2. Therefore, they are not the primary subjects of comparison.

The results are shown in Figure 3 and Figure 4. *FocusLLM with or without parallel* indicates whether we process each chunk either concurrently or sequentially. The findings indicate that: (1) When ample memory resources are available, parallel processing is more efficient for FocusLLM. (2) Although FocusLLM splits long texts into numerous chunks, resulting in a slightly longer inference time compared to the standard approach, it still holds a significant advantage over other long-context methods.

5.4 Chunk Size

We conduct an investigation into the impact of different chunk sizes on performance. In theory, larger chunk sizes, as long as they do not exceed the model's default context length (e.g., 4K for LLaMA-2), are preferable because they allow for processing the memory with a smaller number of forward passes. However, smaller chunk sizes may enable more precise processing.

In experiments, we maintain a total sequence length of 8K, testing the perplexity using different chunk sizes on the same samples of PG19. We select {256, 512, 1024, 2048} as our test sizes. The results are shown in Figure 5. We observe that there is no consistent trend in perplexity as the chunk size increases; it remains relatively stable. This confirms our hypothesis that we can employ larger chunk sizes on models with longer default context lengths (e.g. LLaMA-2-32K). We will explore this direction in our future work.

5.5 Ablation Studies

We employ both Continuation Loss and Reconstruction Loss for the training of FocusLLM. The motivation behind this is to equip the model with the natural language modeling capability while also enhancing its ability to recover information. Ablation

	Hyper Params.	LongBench NarrativeQA	LongBench TREC	∞-Bench Math.Find	∞-Bench En.MC	∞-Bench Retrieve.PassKey
FocusLLM	(2K, 2K)	**18.53**	**65.5**	13.43	**31.00**	**99.32**
Continuation Loss only	(2K, 2K)	17.36	60.5	**13.71**	27.95	1.69
Reconstruction Loss only	(2K, 2K)	17.05	62.0	12.86	26.64	**91.19**
Local Context Size ↓	(1K, 2K)	17.87	63.0	8.86	29.69	**99.32**

Table 4: Investigations into the training loss and local context size of FocusLLM. We present the results for representative tasks from LongBench and ∞-Bench. For instance, NarrativeQA belongs to Single-Doc QA, while TREC relates to Few-shot learning. The Hyper Params is denoted as (local context size, chunk size).

studies as detailed in Table 4, reveal that relying solely on the Continuation Loss enables the model to manage some tasks effectively. Nonetheless, for tasks with substantial dependencies on the preceding context, like HotpotQA and Retrieve.PassKey, the model's efficacy deteriorates. Similarly, while employing the Reconstruction Loss ensures accurate restatement of the preceding context, the lack of generalizability of generating new tokens leads to a considerable decrease in performance. Therefore, the combined use of both loss functions is crucial for enhancing the performance and generalizability of FocusLLM.

We also investigate how the local context size influences performance in the last row of Table 4. As we reduce the local context size from 3.5K to 1K, the performance of most tasks experiences a slight decline. This suggests that candidate tokens cannot fully replace the information within the context.

6 Related Work

6.1 Long-context language models

One research direction involves length extrapolation in transformers (Peng et al., 2023; Jin et al., 2024), where methods like positional interpolation help models adapt to longer sequences (Chen et al., 2023a). However, these techniques often fail to address the distraction issue caused by noisy content within extended texts (Tworkowski et al., 2024). Another research branch focus on modifying the attention mechanism or employing compression techniques to maintain long texts within manageable lengths (Chevalier et al., 2023; Zhang et al., 2024a). For instance, (Xiao et al., 2023) discovered that retaining 'sink tokens' in conjunction with a sliding window can achieve smooth streaming output. (Zhang et al., 2024a) expanded the context dramatically through compression. However, these methods share a common limitation: they cannot utilize information from all tokens.

6.2 Memory-enhanced Model

The integration of memory layers within transformer architectures has become a pivotal strategy for enhancing long-context comprehension (Bertsch et al., 2024; Tworkowski et al., 2024). Common methodologies in memory-enhanced models often employ recurrent strategies that iteratively integrate information from the current window into a persistent memory (Munkhdalai et al., 2024). Another approach is to initially encode the complete long text into memory tokens, which is then queried in to retrieve pertinent information as needed (Xiao et al., 2024). For example, (Yen et al., 2024) employ a small encoder to sequentially encode long text segments, followed by the integration of these encoded chunks into a decoder. However, the drawback of such methods is that the memory length does not extrapolate well, and expanding the memory still incurs substantial computational costs. In contrast, FocusLLM offers superior training efficiency and remains effective on exceedingly long texts.

7 Conclusion

In this work, we introduced FocusLLM, a novel framework that significantly extends the context length of LLMs. The core innovation lies in the parallel decoding strategy, which distribute the burden of understanding long texts across each chunk and effectively aggregating global information. FocusLLM stands out due to its remarkable training efficiency, allowing us to achieve substantial gains in context comprehension with minimal computational and memory cost. Compared to existing methods, FocusLLM not only exhibits superior performance across downstream tasks but also maintains low perplexities when handling extensive texts, up to 400K tokens. We hope FocusLLM can be an inspiring work for the community, driving further exploration of long-context models.

8 Limitations

Our research has certain limitations: (1) Due to hardware constraints, our tests were limited to 400K tokens, which does not represent the upper bound of FocusLLM's capabilities. Future work will explore the full performance potential of FocusLLM and investigate the use of quantization methods to reduce operational costs. (2) While FocusLLM demonstrates exceptional training efficiency, we have observed that training on larger datasets can significantly enhance its generalizability and performance. Therefore, increasing the training data size will be a focus of future research.

References

Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. 2023. Proofpile: A pre-training dataset of mathematical texts.

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

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

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

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

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

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

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Amirkeivan Mohtashami and Martin Jaggi. 2024. Random-access infinite context length for transformers. *Advances in Neural Information Processing Systems*, 36.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

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

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Together. 2023b. Redpajama: An open source recipe to reproduce llama training dataset.

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*.

Lewis Tunstall, Leandr Von Werra, and Thomas Wolf. 2022. Natural language processing with transformers.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

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.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*.

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

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2023. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*.

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

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024a. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024b. ∞bench: Extending long context evaluation beyond 100k tokens. *Preprint*, arXiv:2402.13718.

A Efficiency of FocusLLM

The parallel decoding mechanism of FocusLLM effectively reduces the computational complexity of the standard architecture. Specifically, when dealing with very long sequences, the primary computational burden in the transformer architecture lies in the attention mechanism, which has a complexity of $O(L^2)$, where L represents the total sequence length. By dividing the sequence into n chunks, the complexity within each chunk becomes $O((L/n)^2)$. Therefore, when we process chunks in parallel, the time complexity can be reduced to $O((L/n)^2)$. And the space complexity of n chunks becomes approximately $O((L/n)^2 * n) = O(L^2/n)$. This means that compared to a standard transformer, FocusLLM can reduce the computational complexity to a fraction, $1/n$ or even more of the original theoretically, where n is the number of chunks into which the sequence is divided. In experiments, the longer the sequence length, the more apparent the improvement in efficiency.

B Details of Training Data

We randomly sampled 80K sequences from RedPajama as our training corpus. Table 5 shows the detailed distribution.

Length	3K~4K	4K~6K	6K~8K	Total
Count	30K	16K	34K	80K
Portion	38%	20%	42%	100%

Table 5: Length distribution of training corpus.

C Experimental Details

We primarily conduct experiments on the LLaMA2-7B-Chat model. The additional trainable parameters mentioned in Section 2 amount to only 2B approximately.

Specifically, we conducted training on a Linux server equipped with 8×A100 GPUs, each with 40GB of memory. The training was carried out for 10,000 steps, equivalent to one epoch of the entire training dataset, using a batch size of 8 and a learning rate of 5e-5 with a linear scheduler. To conserve GPU memory, we employed deepspeed's zero2_offload optimizing stage. The training process was completed in approximately 20 hours.

For hyper-parameters, during training, the chunk size was randomly selected from the set {64, 128, 256, 1024, 2048}. For the length of tokens injected into each chunk, we set a default of 512 tokens for inference. And we ensured this length did not exceed the chunk size in the training procedure. As a result, the length of injected tokens was $\min\{512, \textit{chunk size}\}$. For evaluations on the Longbench, we adopt a larger local context size of 3,500 tokens for FocusLLM, consistent with the official setting.

D Details of Benchmarks

D.1 LongBench

LongBench(Bai et al., 2023) includes 14 English tasks, 5 Chinese tasks, and 2 code tasks, with the average length of most tasks ranging from 5K to 15K. In experiments, we only utilize the English tasks. Detailed statistics of the tasks used in our paper are shown in Table 6.

D.2 ∞-Bench

The benchmark (Zhang et al., 2024b) comprises 12 unique tasks, each crafted to assess different aspects of language processing and comprehension in extended contexts. Detailed statistics of the tasks used in our paper are shown in Table 7.

Task	Task Type	Eval metric	Avg len	Language	Sample
HotpotQA	Multi-doc QA	F1	9,151	EN	200
2WikiMultihopQA	Multi-doc QA	F1	4,887	EN	200
MuSiQue	Multi-doc QA	F1	11,214	EN	200
MultiFieldQA-en	Single-doc QA	F1	4,559	EN	150
NarrativeQA	Single-doc QA	F1	18,409	EN	200
Qasper	Single-doc QA	F1	3,619	EN	200
GovReport	Summarization	Rouge-L	8,734	EN	200
QMSum	Summarization	Rouge-L	10,614	EN	200
MultiNews	Summarization	Rouge-L	2,113	EN	200
TriviaQA	Few shot	F1	8,209	EN	200
SAMSum	Few shot	Rouge-L	6,258	EN	200
TREC	Few shot	Accuracy	5,177	EN	200
PassageRetrieval-en	Synthetic	Accuracy	9,289	EN	200
LCC	Code	Edit Sim	1,235	Python/C#/Java	500
RepoBench-P	Code	Edit Sim	4,206	Python/Java	500

Table 6: Detailed statistics of the tasks used in our paper of LongBench.

Task Name	Context	Examples	Avg Input Tokens	Avg Output Tokens
En.MC	Fake Book	229	184.4k	5.3
Code.Debug	Code Document	394	114.7k	4.8
Code.Run	Synthetic	400	75.2k	1.3
Math.Find	Synthetic	350	87.9k	1.3
Retrieve.PassKey	Synthetic	590	122.4k	2.0
Retrieve.Number	Synthetic	590	122.4k	4.0
Retrieve.KV[$\hat{2}$]	Synthetic	500	89.9k	22.7

Table 7: Detailed statistics of the tasks used in our paper of ∞-Bench.

	Activation Beacon
Code Debug	21.32
Math Find	11.71
Math Calc	0.00
Passkey	1.69
Number String	1.69
KV Retrieval	0.00

Table 8: The accuracy of Activation Beacon on ∞-Bench.

E Details of the effective lengths of models in Table 2 and 3

Not all models are capable of processing infinite text lengths. Therefore, we provide a clear explanation of the effective input length for each method in Table 2 and Table 3. Specifically: (i) For models with a finite context length, we truncate the inputs by only preserving the system prompts and the tail of inputs to simulate real-world applications with streaming inputs like (Xiao et al., 2024). For instance, in Table 2, these models include Original (4K), LChat (32K), Vic-16K (16K), Yarn (128K), PI (128K), and NTK (128K). (ii) For other models, including StreamingLLM, InfLLM, LongLlama, CEPE, Activation Beacon, and our FocusLLM, the input can theoretically be of any length. So we input the entire sequence on the two benchmarks.

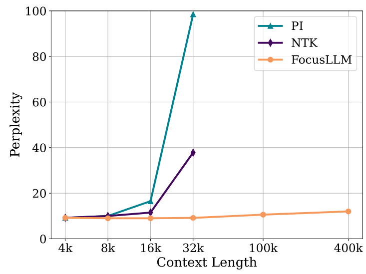

Figure 6: Perplexity on PG19 dataset of FocusLLM compared to methods PI and NTK. FocusLLM can maintain low perplexity even at token counts up to 400K tokens.

F Supplementary Results on ∞-Bench of Activation Beacon

Due to the compression of the context cache, Activation Beacon cannot retain full global information, which hinders its ability to handle tasks that require precise comprehension of the entire text in real-life scenarios, as demonstrated in the results presented in the Table 8.

G Scaling language modeling to 400K context

As shown in Figure 6, FocusLLM maintains a low perplexity even with a context length of 400K. Note that the number of candidate tokens corresponding to 400K is 200, which is far greater than the number of candidate tokens seen during training. This demonstrates that FocusLLM has strong extrapolation capabilities. We can effectively scale to lengths greater than 400K by either using longer sequences during training or by employing a base model with a default context length, which we plan to explore in future work.

H Visualization of Attention Heatmap

We visualized the information carried by candidate tokens when their Key/Value representations are concatenated with the tokens in the local context, and select a few representative heads in Figure 7 and Figure 8. We found that different patterns emerge in Passkey Retrieval and NarrativeQA tasks. The y-axis corresponds to the query representations of tokens in the local context, and the x-axis corresponds to the key representations of candidate tokens combined with the local context tokens. Therefore, the first few columns of the heatmap represent the contribution of candidate tokens to the local context. We made the following interesting observations: **i)** Not all heads in all layers attend to candidate tokens, and higher layers attend to candidate tokens more frequently than lower layers. This is likely because higher layers are more critical for the final representation. **ii)** In Passkey Retrieval task, only one chunk contains passkey information, while the others are noises. As a result, we observe that a single candidate token receives high attention (a single column is highlighted), while other candidate tokens are ignored. **iii)** In NarrativeQA task, the final answer may rely on information from multiple chunks, so we see that many candidate tokens are assigned higher attention weights. In summary, the result indicates that FocusLLM effectively ignores noise and aggregates information from multiple chunks.

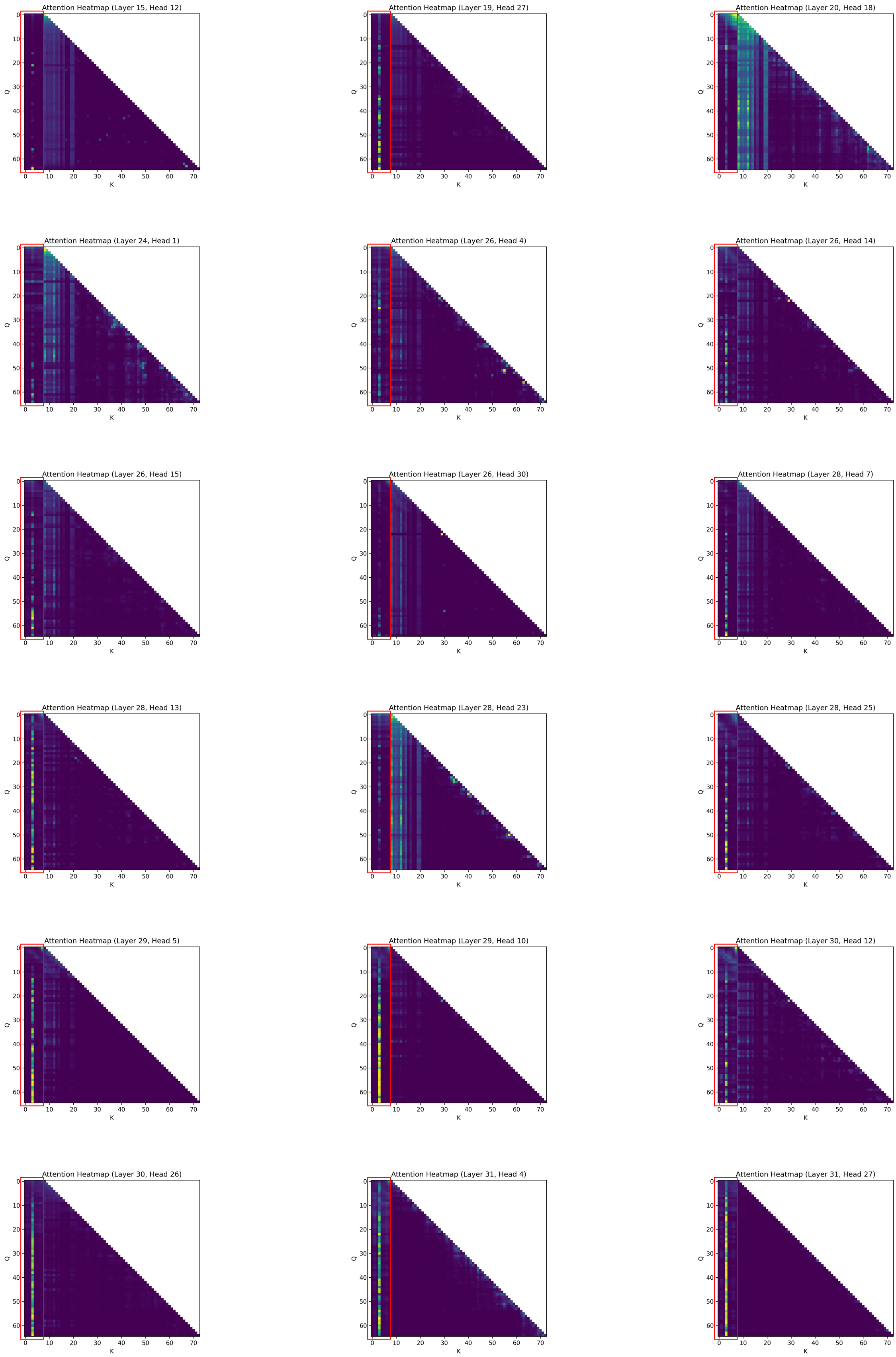

Figure 7: Attention Heamap of Passkey Retrieval task. The first 8 columns, marked by red rectangule lines, represent the attention weights corresponding to 8 candidate tokens. Since only one chunk contains the important passkey information while the others are merely noises, we observe that only a single candidate token receives high attention score (with a single column highlighted). This suggests that FocusLLM can effectively extract important information while discarding irrelevant texts.

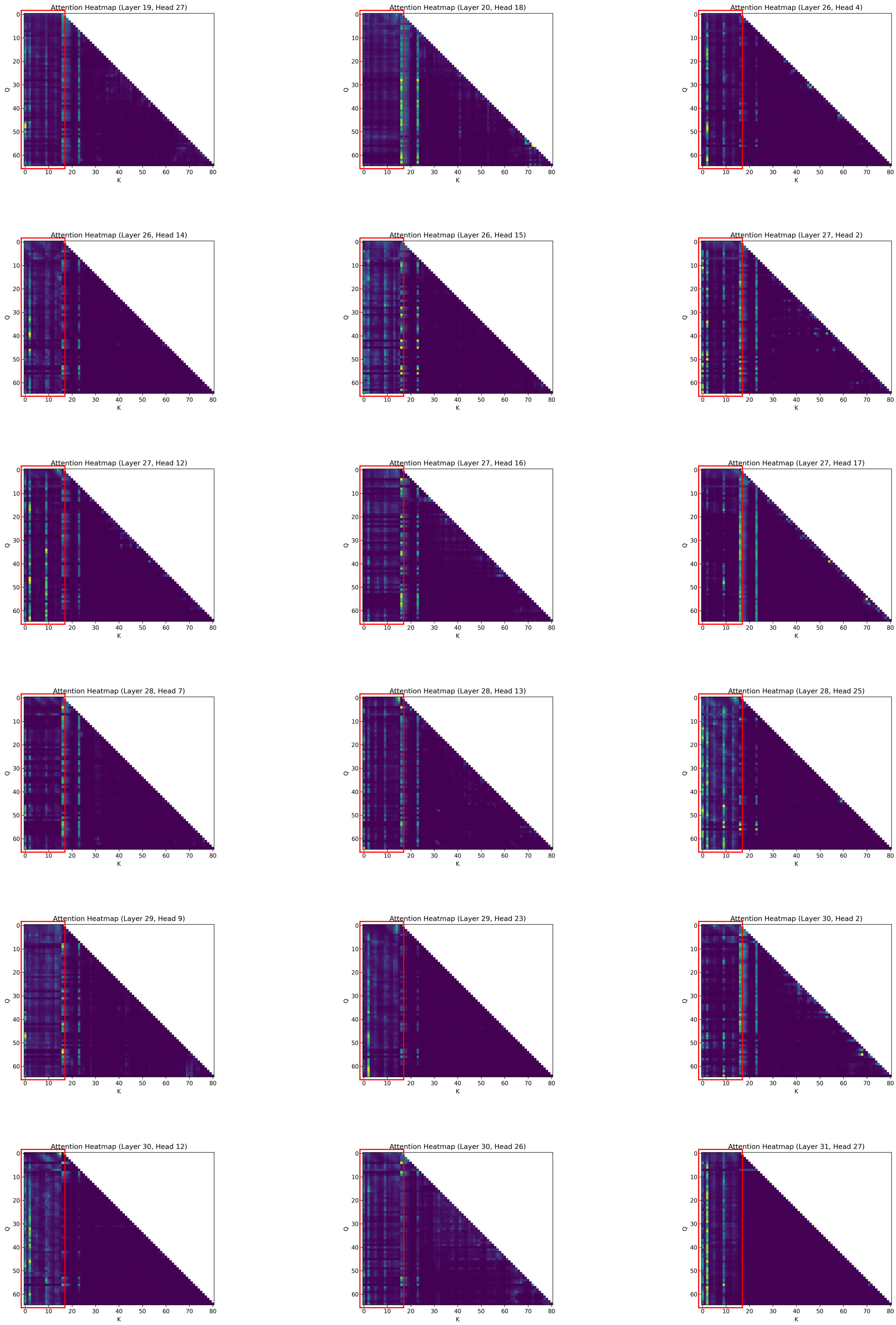

Figure 8: Attention heamap of NarrativeQA in LongBench. The first 15 columns, marked by red rectangle lines, represent the attention weights corresponding to 15 candidate tokens. Since the final answer may rely on information from multiple chunks, we observe that many candidate tokens are assigned high attention weights. This suggests that FocusLLM effectively aggregates information from multiple chunks.