Soaring from 4K to 400K: Extending LLM's Context with Activation Beacon

Peitian Zhang[1,2*] **Zheng Liu**[1†] **Shitao Xiao**[1] **Ninglu Shao**[1,2] **Qiwei Ye**[1] **Zhicheng Dou**[2]
1: Beijing Academy of Artificial Intelligence,
2: Gaoling School of Artificial Intelligence, Renmin University of China
{namespace.pt, zhengliu1026}@gmail.com

Abstract

The utilization of long contexts poses a big challenge for large language models due to their limited context window length. Although the context window can be extended through fine-tuning, it will result in a considerable cost at both training and inference time, and exert an unfavorable impact to the LLM's original capabilities. In this work, we propose **Activation Beacon**, which condenses LLM's raw activations into more compact forms such that it can perceive a much longer context with a limited context window. Activation Beacon is introduced as a plug-and-play module for the LLM. It fully preserves the LLM's original capability on short contexts while extending the new capability on processing longer contexts. Besides, it works with short sliding windows to process the long context, which achieves a competitive memory and time efficiency in both training and inference. Activation Beacon is learned by the auto-regression task conditioned on a mixture of beacons with diversified condensing ratios. Thanks to such a treatment, it can be efficiently trained purely with short-sequence data in just 10K steps, which consumes less than 9 hours on a single 8×A800 GPU machine. The experimental studies show that Activation Beacon is able to extend Llama-2-7B's context length by $\times$100 times (from 4K to 400K), meanwhile achieving a superior result on both long-context language modeling and understanding tasks. Our model and code will be available at the BGE repository[3].

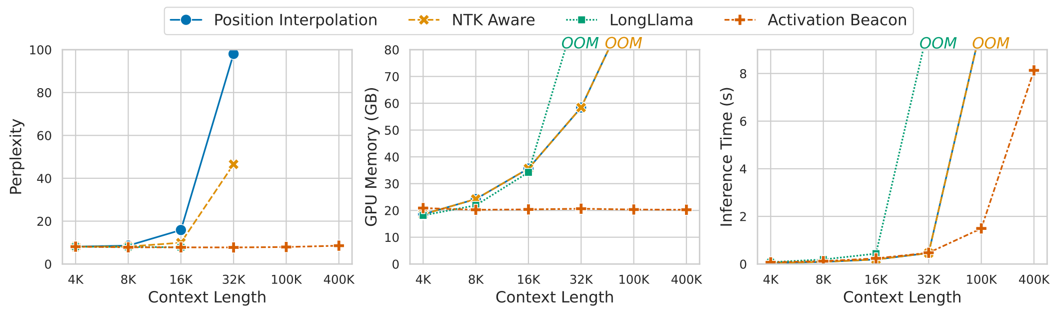

Figure 1: Comparison between Activation Beacon and other context extension methods, including 1) Position Interpolation [5], 2) NTK-Aware Scaled RoPE [1], 3) LongLlama [32]. Activation Beacon leads to a better long-context generation quality with a higher running efficiency (memory, time). The perplexity is measured by sliding windows on PG19 testset [6; 20].

[*]Peitian Zhang and Zheng Liu are the co-first authors
[†]Zheng Liu is the corresponding author
[3]https://github.com/FlagOpen/FlagEmbedding

Preprint. Under review.

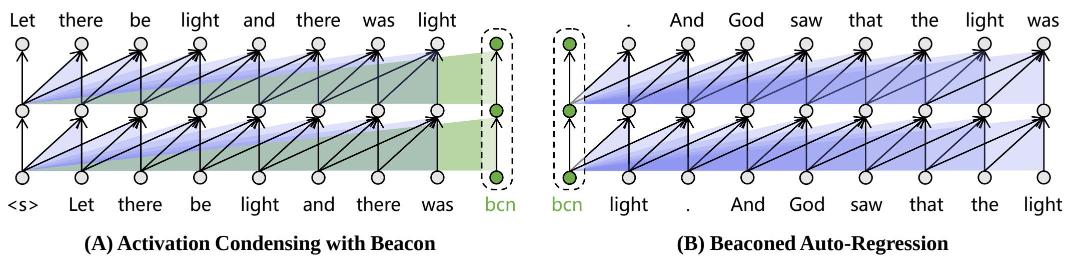

Figure 2: (A) The beacon is used to condense the activations into more compact forms; (B) The condensed activations are streamingly processed within a short sliding window for auto-regression.

1 Introduction

Large language models (LLMs) need to process long contexts to accomplish many important tasks, such as retrieval-augmented generation and in-context learning. However, existing LLMs are typically constrained by fixed context lengths, e.g., 2048 for Llama-1 [29] and 4096 for Llama-2 [30], which is far from enough to handle some real-world scenarios. Although LLMs can be fine-tuned or retrained to establish longer context windows [16; 6; 5; 28; 20; 32; 18], it will result in considerable costs at both training and inference time due to the quadratic computing complexity of self attention. Besides, the continued training on long-sequence data may compromise the LLM's general capability on shorter contexts, which is unfavorable to their practical usability. In light of these challenges, it is desirable to explore new mechanisms, which can not only realize the cost-effective extension of context length for the well-pretrained LLMs, but also be compatible with their existing capabilities.

To address the above problem, we propose Activation Beacon for the extension of LLM's context length (shown as Figure 2). Our method is inspired by an arguably correct hypothesis that the LLM's raw activations are information redundant. As a result, they can be condensed into much more compact forms with very little information loss. On top of such highly condensed activations, the LLM will be able to perceive the information from a vast scope of context even with a very short context window. The above idea shares the common philosophy as sparse attention [8; 3; 38; 12] and context compression [4; 7; 19; 22; 14]. However, it enjoys substantial advantages over the previous methods in terms of effectiveness of context extension (esp. quality of long-context generation and flexibility of supporting diverse context lengths), inference and training efficiency, and compatibility with the existing LLMs and the basic infrastructure thanks to a series of crucial technical designs.

In our work, the LLM's raw activations are condensed by the special tokens, known as *beacons*. The beacons are dispatched to different positions of the context and used to generate the condensed activations for each specific *interval*. For one interval of length L, a team of k beacons are deployed ($L \gg k$), leading to a condensing ratio of α for the raw activations ($\alpha = L/k$). The beacons are parameter efficient because they mostly relies on the LLM's original parameters. Furthermore, the beacons are learned in a post-hoc manner on top of a fixed LLM. Therefore, they can work as a *plug-and-play component*, introducing extended contextual information for the LLM without adversely affecting its existing capabilities on the short context.

The condensed activations from beacons are *streamingly processed* by sliding windows. Particularly, each sliding window is formulated as $[bcn_1, \dots\ bcn_m, x_{m+1}, \dots\ x_n]$, where bcn_i represents the beacon generated by the previous windows and x is an ordinary token in the current window. The size of sliding window is upper-bounded by the maximum context length of the LLM, e.g., 4096 for Llama-2. In this way, the long-sequence data can be processed window-by-window instead of simultaneously, which significantly benefits the running efficiency at both training and inference time.

Activation Beacon can be learned through the auto-regression task: in each sliding window, the generation likelihood of one ordinary token x_i is maximized based on the beacons and its preceding ordinary tokens, i.e., $\max p(x_i|bcn_1, \dots bcn_m, x_{m+1} \dots x_{i-1})$. Considering that an aggressive extension of the context length calls for a large condensing ratio, while a moderate extension just needs a small condensing ratio, we perform random sampling of α during the streaming process. Consequently, the generation can be conditioned on a mixture of beacons with diversified condensing ratios, which substantially contributes to the Activation Beacon's generalization in handling the extension of different context lengths.

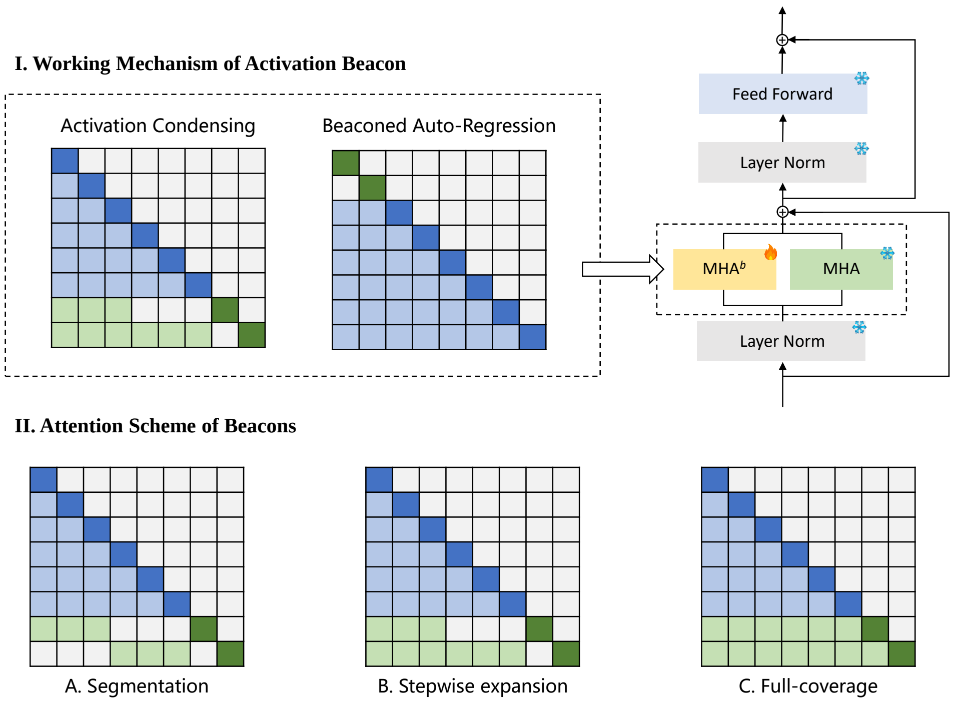

Figure 3: The working mechanism of Activation Beacon (I) and the attention scheme of beacons (II).

Activation Beacon is applied to the Llama-2-7B (chat) model, whose maximum context length is 4K. The training is performed over the mixture of sampled data from RedPajama [10] and LongAlpaca [6] (all training instances are less than 8K), where it merely takes 10K steps (accomplished less than 9 hours on an 8×A800 GPU machine) to reach a strong performance. Notably, it leads to a superior quality of language modeling on the extended context lengths, like 8K, 16K, and 32K, whose result is even better than the finetuned full-attention baselines. It is equally competitive on long context understanding tasks, such as question answering and few-shot learning. Activation Beacon also shows the capability to establish super long contexts, as the context length can be remarkably extended to 100K and 400K.

To summarize, Activation Beacon presents an effective, efficient, compatible, and low-cost (training) method to extend the context length of LLM. It should also be noted that Activation Beacon does not conflict with other alternatives, such as the fine-tuning and retrieval-based methods [2]. In fact, they can complement each other for an even longer extension of the LLM's context.

2 Activation Beacon

2.1 Overview

The LLM generates content by predicting the next token based on its preceding context. Without loss of generality, the prediction can be expressed as: $p(x_n|x_1, ...x_{n-1}; \theta)$, $n \leq L^*$, where θ is the parameters of LLM and L^* is the maximum length of LLM. The existing LLMs are still severely constrained by their maximum context length, which is the major obstacle of utilizing the rich information of the long-sequence data. However, the above constraint can be conquered by a large extent if the LLM's raw activations can be condensed into more compact forms. On top of the highly condensed activations, the LLM will be able to perceive much more information from a vast scope even if the maximum length of the context window stays the same.

2.2 Activation Condensing

When predicting the next token x_n, the LLM queries the activations within the context window, i.e. the keys $K_{:n-1}$ and values $V_{:n-1}$, for the useful preceding information. As introduced, the raw activations are memory consuming and expensive to query, which severely constrain the size of context window. In our work, we introduce the special tokens, namely the beacon, which condenses LLM's raw activations (the blue squares in Figure 3) into more compact ones (the green squares in Figure 3). Consequently, the same context window can intake more information from the previous context, which will benefit the prediction of new tokens. Specifically, we divide the entire context into intervals of length L, where k beacons are dispatched to the end of each interval. In each decoding layer of the LLM, the input hidden states of beacons ($H_b \in \mathbb{R}^{k \times D}$) are transformed to query the raw KV activations within the interval: $\{K, V \mid K \in \mathbb{R}^{L \times D}, V \in \mathbb{R}^{L \times D}\}$, where the condensed activations can be produced. Formally,

$$Q_b \leftarrow H_b W'_Q, \quad K_b \leftarrow H_b W'_K, \quad V_b \leftarrow H_b W'_V \tag{1}$$

$$A \leftarrow \mathrm{softmax}\big(\mathrm{mask}(Q_b\{K \oplus K_b\}^T)\big), \quad V_b \leftarrow A\{V \oplus V_b\}^T, \quad O_b \leftarrow V_b {W'_O}^T. \tag{2}$$

The newly generated KV activations for the beacons, i.e., $K_b, V_b \in \mathbb{R}^{k \times D}$, which leads to a condensing ratio of $\alpha = L/k$ ($k \ll L$). To optimize the condensation's effect, we explore three attention schemes of the beacons, i.e. three implementations of $\mathrm{mask}(\cdot)$, as shown in Figure 3 II: 1) Segmentation, where each beacon can attend to an equally segmented sub-interval of the context. 2) Stepwise expansion, where each beacon can attend to one more sub-interval than its predecessor, and the last beacon can attend to the entire context. 3) Full coverage, where the entire context can be attended by all beacons. The three options are of the same computation cost. However, it's empirically found that the second option, i.e. the stepwise expansion, leads to the optimal performance.

2.3 Beaconed Auto-Regression

The condensed activations from the beacons together with the raw activations from the ordinary tokens are streamingly processed with sliding windows. Each sliding window consists of m beacons from the past context intervals, and the ordinary tokens from the latest context interval. With the above formulation, the next token can be predicted as:

$$p(x_n | bcn_1, ...bcn_m, x_{m+1}, ...x_{n-1}; \theta, \theta_b). \tag{3}$$

Both bcn_* and x_*, are encoded by their relative positions within the sliding window, regardless of their absolute positions in the entire context. Therefore, it is free from modifying LLM's original position encoding scheme. The size of the sliding window is small and is up-bounded by the maximum context length, i.e., $n \ll L^*$, which brings forth high efficiency in terms of both memory and time during training and inference. Nevertheless, it is able to cover $\alpha(m-1)+n$ past tokens on top of the sliding window. When using a large condensing ratio (α) and a large amount of beacons (m), the LLM can access to the information from a broad scope of the context.

2.4 Learning Method

Beacon as a plug-in. Beacon is made up of the following parameters (θ_b): beacon's embedding $bcn.embed$, and beacon's layerwise linear projection matrices $\{W'_Q, W'_K, W'_V, W'_O\}^l$. Beacon reuses other transformer modules from the LLM, i.e., FFN and LayerNorm, which turns out to be the optimal trade-off of effectiveness and training cost. As a result, beacon accounts for less than $1/3$ of the LLM's original size, e.g., 2.1B with the Llama-2-7B (chat) model. Beacon is a plug-and-play module for the LLM. It is learned while the LLM's parameters are fixed. Besides, it is only used to condense the raw activations for the past contexts. Thus, it is able to introduce the long contextual information for the LLM without affecting the LLM's existing capabilities on processing the short context.

Auto-Regression. Beacon is trained by auto-regression, where the next token is predicted based on the condensed activations from beacons and the raw activations from the ordinary tokens. As mentioned in §2.2, a training instance is partitioned into equal-sized intervals. For each interval of length L, the following loss is minimized:

$$\min_{\theta_b} . - \sum_X \sum_{j=1}^{\lceil |X|//L \rceil} \sum_{i=1}^{L} \log p(x_{j,i} | bcn_1, ...bcn_{m_j}, x_{j,1}, \ldots, x_{j,i-1}; \theta, \theta_b). \tag{4}$$

where $x_{j,i}$ is the i-th token in the j-th interval of X, m_j stands for the number of beacons generated before the jth interval whose value depends on the condensing ratio of each preceding interval (α_i).

Stepwise sampled condensing ratio. The training is performed on short-sequence data, i.e. $1024 < |X| < 8192$, where the majority of training instances are less than 4096 (see Table 1). Therefore, we are able to achieve a superior training efficiency. To generalize the beacon for longer and diverse context lengths, e.g., 16K, 32K, and beyond, the auto-regression needs to be conditioned on different amount of beacons of diversified condensing ratios. For this purpose, we perform stepwise sampling at each interval where the condensing ratio is randomly chosen from a large scope: $\alpha_i \sim$ [2, 4, 8, ... 128]. This operation will dramatically diversify the condensing ratios and the amount of beacons where the auto-regression is conditioned.

3 Experiment

Our experiments are performed for the exploration of the following issues. 1) Activation Beacon's impact on the long-context generation capabilities (measured by Perplexity). 2) Activation Beacon's impact on the long-context utilization capability (reflected by tasks like long document QA and summarization). 3) Activation Beacon's impact on efficiency in terms of GPU memory and inference time. 4) The individual contribution of different technical factors.

3.1 Settings

Implementation. Our method is applied to Llama-2-7B (chat) [30] for empirical studies. It is trained with a mixture of 80K sampled data from RedPajama [10] and LongAlpaca [6] (70K from RedPajama and 10K from LongAlpaca, respectively). The sequence length of each sample is between 1024 and 8192. The statistics of our training data is reported in Table 1. We use a single 8×A800 GPU machine for training. The training is performed for 10,000 steps (one epoch of the whole training data) with a batch size of 8 and a learning rate of 5e-5 using the linear scheduler. The length of each context interval is set to 1024. The condensing ratio is sampled from {2, 4, 8, 16, 32, 64, 128} during training. As introduced, Llama's own parameters are fixed throughout the training process.

Length	1K∼2K	2K∼4K	4K∼6K	6K∼8K	Total
Count	38K	23K	6K	13K	80K
Portion	47%	29%	8%	16%	100%

Table 1: The length distribution of the training data. The average length of all training data is 3180.

Baselines. The following types of baselines are chosen for comparison (all based on the LLaMA-2-7B (chat) model unless otherwise specified). 1) The basic method, i.e. LLaMA-2-7B (chat) [29] with 4096 context length. 2) The fine-tuning free methods, including Positional Interpolation (PI) [5], the NTK-Aware Scale ROPE (NTK) [1], and StreamingLLM [36]. 3) The finetuned full-attention methods, including LongChat-32K [16], LongAlpaca-16K [6], YaRN-128K [20]. 4) The fine-tuned methods with adapted architectures for long context, including AutoCompressor-6K [7] and LongLlama [32] (based on CodeLlama [24]). We enable FlashAttention-2 [11] to accelerate self-attention computation and save GPU usage for all the baselines. At the present time, Activation Beacon is incompatible with FlashAttention-2 due to its utilization of the customized attention scheme; thus, we use the scaled dot product attention (sdpa) from PyTorch [17] for acceleration.

3.2 Main Results

3.2.1 Long Context Language Modeling

The experiment on long context language modeling is performed with three datasets: PG19 [22], Proof-Pile [40], and CodeParrot [31]. Specifically, for PG19, we use its entire test set with 100 books. For Proof-Pile, we extract the arxiv papers from the test set and sample 100 texts for evaluation. For CodeParrot, there is not pre-defined test set. Following previous studies [25; 39], we first concatenate code from the same repository to form long sequences, then we sample 100 sequences for evaluation. The perplexity is computed with sliding window of size 2048 [21].

The evaluation results are reported in Table 2, where Activation Beacon leads to a superior long context language modeling performance. First of all, it not only outperforms the Llama-2-7B baseline but also results in a notably improved performance than the fine-tuning free methods. It should be also noted that with the extension of context length from 4K to 32K, the language modeling performance can be gradually improved by Activation Beacon, indicating that the expanded information from the

Method	PG19				Proof-Pile				CodeParrot			
	4K	16K	32K	100K	4K	16K	32K	100K	4K	16K	32K	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

Table 2: Sliding window perplexity of different context window extension methods on PG19, Proof-Pile, and CodeParrot. Activation Beacon successfully extends the context window of Llama-2-7B model to sequences much longer than ones seen during training.

longer context can be effectively utilized. By comparison, the language modeling performance is decreased with other fine-tuning free methods. Most of them become ineffective after the context length goes beyond 32K.

Secondly, Activation Beacon's performance is comparable to or even better than the finetuned full-attention methods. This result is remarkable knowing that Activation Beacon runs with a much higher efficiency (to be analyzed in Section 3.3). Although there are cases where some of the finetuned full-attention baselines achieve even better performances, their empirical advantages may not be fully resulted from the introduction of long contextual information. For example, YaRN-128K's performance is siginificantly improved over Llama-2-7B at the context length of 4K, and so is the case with LongChat-32K on Proof-Pile and CodeParrot. It is worth noting that the update of the LLM's original parameters are not always favorable because it may not be well generalized to many other scenarios. By comparison, our method is simply a plug-and-play module where the long contextual information can be introduced without affecting the LLM's existing capabilities.

Thirdly, Activation Beacon is able to achieve a much longer extension of the context than the rest of the methods. Particularly, it maintains a quality generation performance after the context length is extended to 100K, where most of the baselines become either ineffective or out-of-memory (OOM). In fact, Activation Beacon is still effective even after the context length is further extended to 400K (see Figure 1), which means a 100× extension of Llama-2-7B's maximum context length. In addition, unlike many alternative methods like fine-tuning, Activation Beacon does not require any long-sequence training data to acquire such a super long context capability.

3.2.2 More Long Context Tasks

Method	Single-Doc QA	Multi-Doc QA	Summarization	Few-Shot	Code
Llama-2-7B	24.90	22.60	24.70	60.00	48.10
PI	18.98	17.16	25.03	49.43	52.73
NTK	23.21	23.34	24.40	59.29	49.28
StreamingLLM	21.47	22.22	22.20	50.05	48.00
AutoCompressor-6K	13.22	10.61	14.00	15.72	23.62
YaRN-128K	24.03	24.11	19.82	60.00	62.73
LongChat-4K	28.14	21.88	26.59	62.06	52.77
LongChat-32K	**31.58**	23.50	26.70	**64.02**	54.10
LongAlpaca-4K	26.81	24.44	26.93	62.92	55.15
LongAlpaca-16K	28.70	28.10	**27.80**	63.70	56.00
LongLlama	30.12	16.37	24.19	60.31	**66.05**
Activation Beacon	27.14	**28.28**	25.15	60.72	57.83

Table 3: Evaluation of different methods on LongBench. Activation Beacon performs on par with the fine-tuned full-attention baselines.

We further study the five real-world tasks from LongBench [2], including single-doc QA, multi-doc QA, summarization, few-shot learning, and code completion, where the experiment result on each task is reported in Table 3. We also evaluate the topic retrieval task [16], whose result is shown in

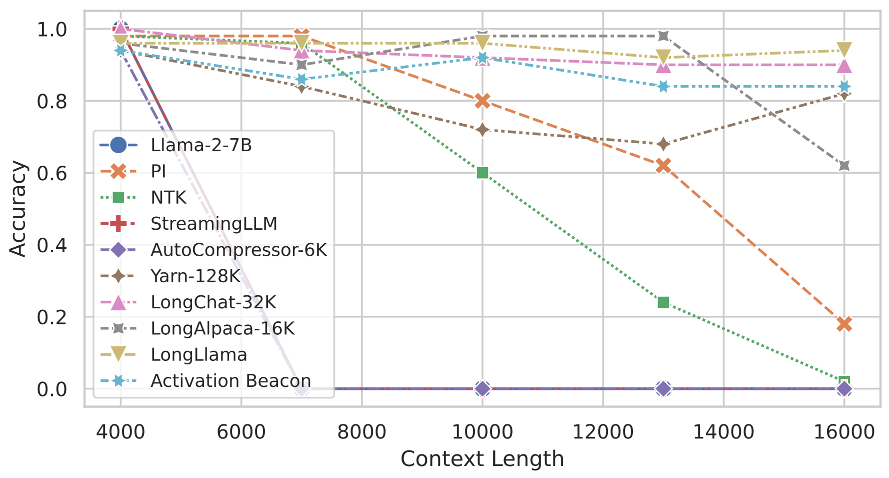

Figure 4: The evaluation of topic retrieval accuracy at different context lengths. Activation Beacon achieves a similar performance as the fine-tuned methods, like LongChat-32K and LongAlpaca-16K.

Figure 4. Similar to our previous observation on long context language modeling, Activation Beacon leads to a notable improvement over Llama-2-7B and the fine-tuning free baselines. Meanwhile, it reaches a comparable performance with the finetuned full-attention methods. Because a large portion of the evaluation samples can be (almost) covered by the 16K or 32K context window, the finetuned full-attention methods indeed set a high bar for this benchmark. However, knowing that the fine-tuning operation will change the LLM's original parameters, it is still interesting to investigate where does the empirical advantage of the finetuned methods come from. To figure out this problem, we benchmark the performance of LongChat-32K and LongAlpaca-16K at the context length of 4K, where they use the same information as the Llama-2-7B baseline. Interestingly, both methods result in a substantial improvement over Llama-2-7B on every task. Especially for summarization, where both methods are already sufficiently strong at 4K, yet little extra improvements are made with the further extended context window. By comparison, Activation Beacon inherits Llama-2-7B's performance at the context length of 4K, where its performance gain over Llama-2-7B fully comes from the extended context. In this sense, its impact on introducing long contextual information can still be no inferior to the ones from the finetuned methods in the corresponding situations.

3.3 Efficiency Analysis

Method	**GPU Memory (GB)**					**Inference Time (s)**				
	4K	8K	16K	32K	100K	4K	8K	16K	32K	100K
LongChat-32K	18.5	24.2	35.6	58.4	OOM	0.045	0.089	0.191	0.460	OOM
StreamingLLM	19.9	19.9	19.9	19.9	19.9	–	–	–	–	–
AutoCompressor-6K	17.7	22.6	32.3	51.7	OOM	0.087	0.134	0.224	0.478	OOM
LongLlama	18.2	21.9	34.2	OOM	OOM	0.079	0.190	0.436	OOM	OOM
Activation Beacon	20.8	20.3	20.5	20.4	20.3	0.071	0.121	0.237	0.473	1.494

Table 4: Evaluation of inference time and GPU memory usage. Both metrics are measured by the average value of 100 forward passes (FlashAttention-2 is enabled for LongChat).

Method	**Time (Hour)**	**Memory (GB)**
LongAlpaca-16K	20.8	57.1
Activation Beacon	9.0	55.9

Table 5: Comparison of training time and GPU memory cost between LongAlpaca-16K (8xA100 GPUs) and Activation Beacon (8xA800 GPUs).

We evaluate the running efficiency at the inference time in terms of time cost and GPU memory usage, whose results are reported in Table 4. Compared with LongChat (full-attention) and LongLlama, Activation Beacon enjoys a much smaller GPU memory usage at the long context lengths. Activation Beacon and StreamingLLM result in a similar memory cost because both methods are based on sliding windows. As for the inference time, Activation Beacon is faster than LongLlama, but slower than LongChat when the context length is short. This is because Activation Beacon is streamingly processed while LongChat is fully parallel[4]. However, Activation Beacon is able to gradually catch up when the context length gets longer, as its time cost is linear to the context length. It will ultimately become much faster than the full-attention methods if the context length is extended long enough. Finally, we also compare our training cost with LongAlpaca, which is featured for its high training efficiency (shown in Table 5). Under a similar hardware condition (8×A800 GPUs vs. 8×A100 GPUs), the training of Activation Beacon can be accomplished in just 9 hours, which is even faster than the reported time cost of LongAlpaca-16K with S^2-attn[5].

3.4 Ablation Studies

Factor	Setting	PG19			Single-Doc QA
		8K	16K	32K	
Attention Scheme	Segmentation	8.72	8.67	8.65	21.30
	Full coverage	8.66	8.63	8.66	25.59
	Stepwise expansion*	8.55	8.54	8.56	27.14
Condensing Ratio	Monotonous ($\alpha = 4$)	8.53	8.56	$> 10^2$	17.85
	Instance-wise sampling	8.73	8.68	8.67	22.76
	Step-wise sampling*	8.55	8.54	8.56	27.14
Data Composition	RedPajama only	8.54	8.54	8.55	24.02
	RedPajama+LongAlpaca*	8.55	8.54	8.56	27.14

Table 6: The impact of different technical factors, including the attention schemes of beacons, the condensing ratio, and the composition of training data. The default setting marked by *.

We perform ablation studies to evaluate the impact from different technical factors, including the attention scheme of beacons (2.2), the sampling strategy of condensing ratio (2.4), and the composition of training data. The experiment results are shown in Table 6.

First of all, We can find that the attention scheme exerts a substantial impact to Activation Beacon's performances on both long-context language modeling (PG19) and long context understanding (Single-Doc QA). The stepwise expansion works with the gradually expanded attention scope. Therefore, it enables the beacons to acquire different levels of local and global visibility of each context interval, which notably improves the empirical performance over the other two options.

Secondly, the sampling of condensing ratio is another influential factor. In this place, we compare two alternative strategies. The instance-wise option samples one condensing ratio for all context intervals of each training instance X (from the same scope as the step-wise method, i.e. [2, 4, 8, 16, 32, 64, 128]). While the monotonous option makes use of one constant condensing ratio of 4 (which can support a up-to 16K context length). We can observe that the step-wise sampling strategy, which introduces the most diversified condensing ratios for auto-regression, results in the strongest empirical performance for every evaluated context length.

Thirdly, we can also observe that only using RedPajama as the training data already leads to a competitive performance on both evaluation tasks. The introduction of more training data from LongAlpica contributes little to the language modeling task. However, it substantially improves the long context understanding performance on Single-Doc QA.

4 Related Works

We discuss the following research works which are devoted to the extension of LLM's context length.

First of all, a large body of methods have been proposed to increase the size of LLM's context window. For example, ALiBi [21] leverages linear-decaying attention biases to achieve the extrapolation of

[4]StreamingLLM is extremely slow due to its current stepwise implementation. However, it should be able to achieve a similar time efficiency as Activation Beacon with improved implementation in the future.

[5]https://openreview.net/forum?id=6PmJoRfdaK

position encoding. Similarly, methods like Position Interpolation [5], NTK-Aware scaling [1] and ReRoPE [26] make progresses on top of RoPE [27], which enables the LLM to handle unseen positions at the inference time. Although such methods can be directly applied to the well-trained LLM, it is usually helpful to perform continual fine-tuning such that the extended context can be better utilized [20]. The fine-tuning with long-sequence data is expensive. Therefore, people have investigated several new methods to reduce the training cost. For example, LongLora [6] proposes S^2-Attn and leverages LoRA for the cost-effective training; while PoSE [41] uses skip-wise position indices to train LLMs on 2K context length as a simulation of 128K. However, the fine-tuning operation will still be extremely expensive if super long context is presented. The fine-tuned LLM still relies on full-attention, where the inference cost will be also unaffordable even with acceleration techniques, like FlashAttention [11]. Finally, the fine-tuning operation may impair the LLM's existing capacity on short contexts [20]. By comparison, our method serves as a drop-in module which is fully compatible with the existing LLM. Besides, it also enjoys much higher efficiency at both training and inference time.

The quadratic complexity of self attention is a major bottleneck to achieve long contexts using transformers. Therefore, many previous works aim to address this problem by using sparse attention [8; 3; 38; 12] or approximate attention computation [15; 33; 9; 23]. However, there are threefold challenges about these methods as analyzed in [36]: the requirement of customized GPU kernels for specific variants of matrix-matrix multiplication, the dependency on global attention patterns which are unsuitable for autoregressive language models, and the incompatibility with the well-pretrained models. In contrast, our method is free from the above constraints, which can work as a drop-in component for the extension of the LLM's context.

Another line of research seeks to offload the long context to external memory devices and retrieve the useful part from it as the working context. The retrieved data can be either the chunked input [37; 39] or the cached KV activations, e.g., Memorizing Transformers [35] and LongMem [34]. The similar idea has been further extended by many recent works. For example, Landmark Attention [18] uses a special token to represent a chunk of activations, which enables more efficient computation of retrieval. Focused Transformers [32] proposes to use contrastive training which improves the discrimination of relevant keys from the cached data. The retrieval-based methods share a similar spirit as our method in the sense of condensing the context, but tackle the problems from a different perspective.

Finally, it is also plausible to introduce long contexts on top of sliding windows. For example, StreamingLLM [36] and LM-Infinite [13] are able to achieve an infinite context by only maintaining the activations for the very first and the latest tokens. However, they are not able to leverage the rich information within the long context because the portion beyond the sliding window will be ignored. Although some other works make use of the summarized or compressed memory to mitigate such a problem [4; 7; 19; 22; 14], it will still incur a substantial loss of long-contextual information given the existing memorization mechanisms. By comparison, our method enjoys both the high running efficiency because of its streaming mechanism and the high effectiveness of utilizing the long context thanks to its high-quality condensing of the activations.

5 Conclusion

In this work, we introduce Activation Beacon for the extension of LLM's context length. Activation Beacon condenses the LLM's raw activations into more compact forms, enabling the LLM to perceive a vast context with a limited context window. As a plug-and-play component for the LLM, it brings in long contextual information while fully preserving the LLM's existing capabilities on short contexts. When dealing with long-sequence data, it resorts to sliding windows for streaming processing, which leads to a superior working efficiency at both inference and training time. With the diversely sampled condensing ratios, it can be effectively learned to support the extensions for a wide scope of context lengths based on short-sequence training data. The experimental study verifies Activation Beacon as an effective, efficient, compatible, low-cost (training) method to extend the context length of LLM.

References

[1] Ntk-aware scaled rope, 2023.

[2] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

[4] Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Scaling transformer to 1m tokens and beyond with RMT. *CoRR*, abs/2304.11062, 2023.

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

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

[7] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics, 2023.

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

[9] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[10] Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023.

[11] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691, 2023.

[12] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1, 000, 000, 000 tokens. *CoRR*, abs/2307.02486, 2023.

[13] Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. *CoRR*, abs/2308.16137, 2023.

[14] Xinting Huang and Nora Hollenstein. Long-range language modeling with selective cache. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 4838–4858. Association for Computational Linguistics, 2023.

[15] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[16] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can open-source llms truly promise on context length?, June 2023.

[17] Christian Puhrsch Michael Gschwind, Driss Guessous. Accelerated pytorch 2 transformers. `https://pytorch.org/blog/accelerated-pytorch-2/`, 2023.

[18] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

[19] Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. *CoRR*, abs/2304.08467, 2023.

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

[21] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[22] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[23] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22470–22482, 2021.

[24] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[25] Ohad Rubin and Jonathan Berant. Long-range language modeling with self-retrieval. *CoRR*, abs/2306.13421, 2023.

[26] Jianlin Su. Rectified rotary position embeddings. `https://github.com/bojone/rerope`, 2023.

[27] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864, 2021.

[28] Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. *arXiv preprint arXiv:2212.10554*, 2022.

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

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

[31] Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. Natural language processing with transformers, 2022.

[32] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*, 2023.

[33] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020.

[34] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *CoRR*, abs/2306.07174, 2023.

[35] Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

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

[37] Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets long context large language models. *CoRR*, abs/2310.03025, 2023.

[38] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[39] Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. Retrieve anything to augment large language models. *CoRR*, abs/2310.07554, 2023.

[40] Bartosz Piotrowski Zhangir Azerbayev, Edward Ayers. Proof-pile. `https://huggingface.co/datasets/hoskinson-center/proof-pile`, 2022.

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