Extensible Embedding: A Flexible Multipler For LLM's Context Length

Ninglu Shao[1,2†] **Shitao Xiao**[1†] **Zheng Liu**[1†] **Peitian Zhang**[1,2]

1: Beijing Academy of Artificial Intelligence

2: Gaoling School of Artificial Intelligence, Renmin University

rainym00d@163.com stxiao@baai.ac.cn zhengliu1026@gmail.com

Abstract

Large language models (LLMs) call for extension of context to handle many critical applications. However, the existing approaches are prone to expensive costs and inferior quality of context extension. In this work, we propose **Extensible Embedding**[1], which realizes high-quality extension of LLM's context with strong flexibility and cost-effectiveness. Extensible embedding stand as an enhancement of typical token embedding, which represents the information for an extensible scope of context instead of a single token. By leveraging such compact input units of higher information density, the LLM can access to a vast scope of context even with a small context window. Extensible embedding is systematically optimized in architecture and training method, which leads to multiple advantages. 1) High flexibility of context extension, which flexibly supports ad-hoc extension of diverse context lengths. 2) Strong sample efficiency of training, which enables the embedding model to be learned in a cost-effective way. 3) Superior compatibility with the existing LLMs, where the extensible embedding can be seamlessly introduced as a plug-in component. Comprehensive evaluations on long-context language modeling and understanding tasks verify extensible embedding as an effective, efficient, flexible, and compatible method to extend the LLM's context.

1 Introduction

Large language models (LLMs) need to process long-sequence data in order to accomplish many critical tasks, like RAG and long-doc reading comprehension. Unfortunately, the existing LLMs are limited by their context windows, which are far from enough to fully cover the input data in corresponding scenarios. To overcome this limitation, people resort to fine-tuning to extend the LLM's context window (Chen et al., 2023b; Dacheng et al., 2023; Peng et al., 2023). Despite the popularity in practice, the fine-tuning approaches will lead to huge training and inference costs. Besides, the fine-tuning over long-sequence data is likely to impair the LLM's original capability on shorter contexts, which is unfavorable to the practical usage. Although there are other alternative ways to establish long contexts, e.g., sparse attention (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020), stream processing (Xiao et al., 2023a; Han et al., 2023), retrieval (Xu et al., 2023; Wu et al., 2022; Tworkowski et al., 2023), the existing solutions are prone to problems, like inferior extension quality or incompatibility with the existing LLMs.

It's usually believed that the size of context window, e.g., 4096 for LLaMA-2 (Touvron et al., 2023), is equivalent to the maximum of tokens the LLM can perceive. However, we challenge this common belief by arguing that the size of context window is just a constraint of the input units instead of the limit of context the LLM can perceive. Based on this argument, we propose a new method called **Extensible Embedding** to facilitate the utilization of long context for LLMs. It stands as an enhancement of typical token embeddings, which is used to represent the information for an extensible scope of context, e.g., multiple words or a sentence. Therefore, it can possess a much higher information density than token embeddings. On top of such compact form of representations, the LLM will be able to access to the information from a vast context with its original context window.

The extensible embedding is realized by a compact model architecture. We employ a lightweight model, namely *extensible embedder*, to transform the input into output embeddings. Then we adopt another *down-scaling function*, which down samples the output embeddings by a factor of k (e.g., $k = 32$). In other words, one out of k output embeddings are sampled as an extensible embedding. Notably, the down-scaling can be conducted with

Could be residual embedding

†. Co-first and co-corresponding author.

1. https://github.com/FlagOpen/FlagEmbedding

an arbitrary scaling factor and sampling scheme at the inference time. Thus, it contributes to a high flexibility of practical usage, which enables the ad-hoc extension of different context lengths.

The extensible embedder is learned through the *two-stream auto-regression (AR)* tasks, where each training sample is processed by two passes of feed-forward. In the first pass, the extensible embeddings are generated and cached for each training sample. In the second pass, the next tokens are predicted based on the extensible embeddings of their preceding contexts. Based on such a tailored for of auto-regression, comprehensive training losses can be derived from all tokens within each training sample. Therefore, it brings forth two benefits: on one hand, the extensible embeddings can be learned to assist the LLM's generation directly, which is well aligned with the downstream LLM's working process; on the other hand, it results in an exceptional sample efficiency, which enables the model to be effectively trained with a small amount of data.

The training process is performed with the downstream LLM's parameters fixed all the time. Thus, the extensible embeddings can work as a *plug-in module*, which brings extended contextual information without compromising the LLM's original performance with short contexts. Interestingly, we also observe the strong but unexpected compatibility of extensible embedding beyond its downstream LLM. In particularly, the well-trained extensible embeddings for one LLM can be effectively applied to other fine-tuned derivatives of its downstream LLM without any further adaptation. Such a property suggests the extensible embedding's potential as a versatile method for the context extension across a family of closely related LLMs.

We initialize the extensible embedder with the first 8 layers of LLaMA-2-7B model (Touvron et al., 2023), where it is trained to extend the context for another downstream LLaMA-2-7B model. With just 100K training samples from RedPajama (Computer, 2023) and LongAlpaca (Chen et al., 2023b), the extensible embedder is able to achieve a superior capability in context extension. Notably, it enables the extension of LLaMA-2-7B (4K) over *100K*, while producing superior performances on both long-context language modeling and understanding tasks. Besides, by applying to other fine-tuned derivatives of LLaMA-2-7B with larger context windows, e.g., LongChat-32K (Dacheng et al., 2023), it can further enable high-quality generation with super-long contexts over *1 million tokens*.

To summarize, this work is highlighted for the following contributions. 1) Extensible embedding presents a simple but effective method, which establishes a long context for the LLM based on compact representation of the input. 2) The tailored model architecture facilitates superior and flexible extension for different context lengths, and 3) the sample-efficient two-stream AR task enables the cost-effective training of the model. 4) Comprehensive experiments verify extensible embedding as an *effective*, *efficient*, *flexible*, and *compatible* method for the extension of LLM's context.

2 Extensible Embedding

2.1 Framework

The workflow of extensible embedding is shown as Figure 1, where a long-sequence input X (e.g., a long document of 16K tokens) can be utilized by a LLM (e.g., LLaMA-2) with a short context window (4K). Firstly, the input X is partitioned into chunks: $\{X_1, ...X_N\}$. The chunk length L_i is set as the maximum window size of the extensible embedder, e.g., $L_i = 4096$ with LLaMA-2, where the coherence of context can be best preserved. Secondly, each chunk is transformed by the extensible embedder into its output embeddings. The output embeddings are down-scaled by the scaling factor k (e.g., $k = 32$), where L/k extensible embeddings (denoted as EX) are produced as the compact representation of the input. Finally, the new tokens are predicted conditioned on the extensible embeddings from the preceding chunks and the normal token embeddings within the recent context.

2.2 Embedding Generation

The typical token embedding, which is corresponding to each individual token, is information sparse. In contrast, the extensible embedding is used to represent an extensible scope of context, e.g., multiple words or sentences, which possess a higher information density. We employ a language model as the embedder (LM_{ex}), which transforms the input $X_i : \{x_{i,1}, ...x_{i,L}\}$ into output embeddings O_i:

$$O_i : \{o_{i,1}, ...o_{i,L}\} \leftarrow \text{LM}_{ex}(x_{i,1}, ...x_{i,L}; \theta_{ex}).$$

On top of an expressive embedder, each output embedding can serve as a high-quality representation for its preceding context, i.e. $o_{i,j}$ for $x_{i,1}, ...x_{i,j}$. To acquire the compact representation for the entire input, the output embeddings are further down-scaled

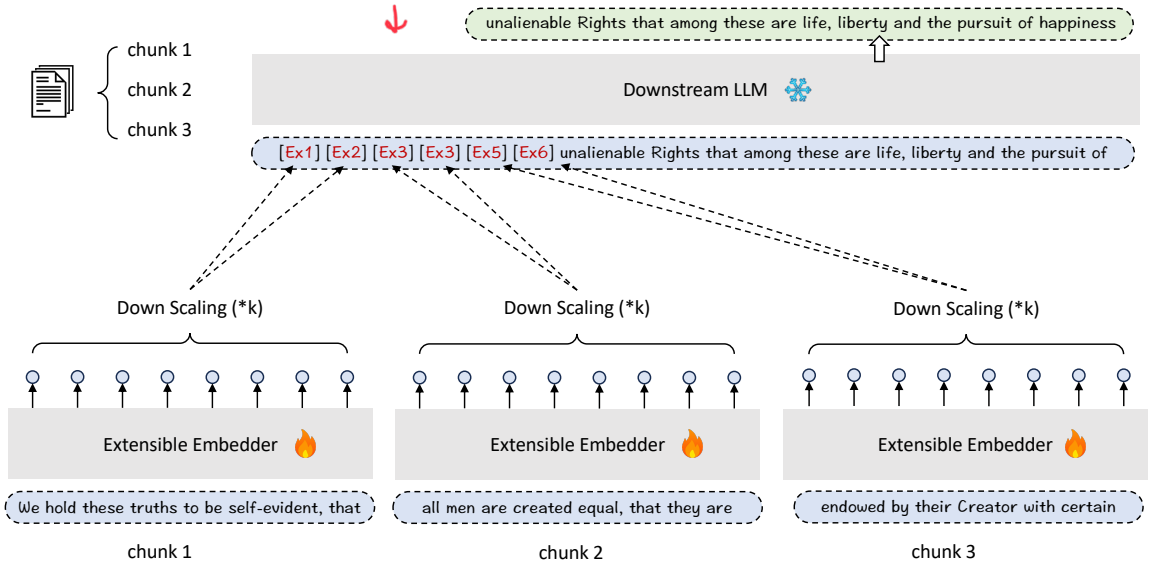

Figure 1: **Framework**. The input data is partitioned into chunks. Each sub-sequence is transformed and down-scaled as extensible embeddings. The new tokens are predicted based on the extensible embeddings from preceding chunks and the token embeddings in the same chunk. The extensible embedder is learned with a fixed downstream LLM.

by the scaling factor k, where m ($m = L/k$) extensible embeddings ($ex_{i,*}$) are generated for X_i:

$$\{ex_{i,1}, ..., ex_{i,m}\} \leftarrow \text{DownScale}(\{o_{i,1}, ...o_{i,L}\}).$$

There can be many alternative ways to realize the functionality of down-scaling, where arbitrary pooling functions along the sequence dimension can be applied. In our work, we simply down-scale the output embeddings through the strided sampling, where the last embedding in every k steps is chosen, i.e., $ex_{i,j} \leftarrow o_{i,k\times j}$. On one hand, such a simple scheme is easy to realize on top of the existing LLM's architecture, and it produces the optimal empirical performance in the downstream tasks. On the other hand, it leads to a high flexibility of usage, where the context can be extended by an arbitrary scaling factor by simply adjusting the downs-sampling rate (k) at the inference time.

2.3 Learning Method

The extensible embeddings are learned by the auto-regression (AR) tasks, where the loss is minimized for the prediction of next tokens conditioned on the extensible embeddings from the preceding context. The auto-regression can be simply performed by having the long context transformed into extensible embeddings and predicting the last few tokens within one training sample, e.g., predicting the answer to a question based on the extensible embeddings of a long document. However, the naive method will be limited by its inferior training effect, because the long context accounts for the majority of computation cost whereas no prediction loss can be produced from it.

In our work, we propose **two-stream AR** which trains the model with optimized sample efficiency (Figure 2). In the first pass of inference, the extensible embeddings are generated for the entire context. For example, with a chunk size of 3 and an scaling factor of 3, the input data $X = \{x_1, ...x_{15}\}$ is transformed into the extensible embeddings $\{ex_{1,1}, ex_{2,1}, ex_{3,1}, ex_{4,1}\}$ (the last chunk is exempted). In the second past, each single token within the long context is streamingly predicted by chunks. Particularly, the prediction is made conditioned on the extensible embeddings from the previous chunks and the preceding normal token embeddings within the same chunk. Formally,

$$\min_{\theta_{ex}} \sum_X \sum_{i>1} \log P(x_{i,j} | ex_{1,1}, ...ex_{i-1,k}, x_{i,1}, ...x_{i,j-1}; \theta, \theta_{ex}).$$

For example, x_6 is predicted based on ex_1 (representing x_{1-3}) and x_4. Crucially, the chunk size of training is made much smaller than the LLM's window size (e.g., 512), where the prediction of new tokens can mostly rely on the contextual information offered by the extensible embeddings. Thanks to the above processing, the prediction loss can be comprehensively derived from the each training sample, which enables the model to be effectively learned from a small amount of data. We also randomly sample the extension ratio k from a candidate scope (e.g., [2, 4, 8, 16, 32]) for each training

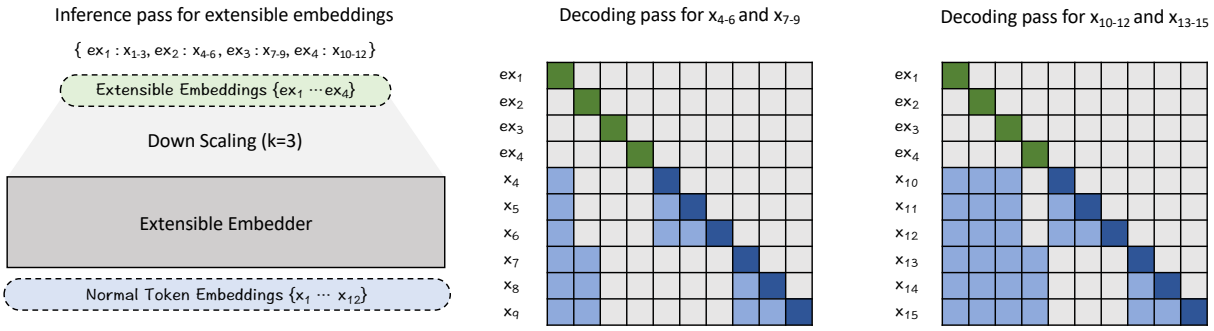

Figure 2: Two-Stream AR. In the first pass, the normal embeddings are transformed into extensible embeddings (with a scaling factor $k = 3$). In the second pass (window size 10, chunk size 3), the auto-regression is accomplished in two sliding steps: the x_{1-3} and x_{4-6} predicted in the first step, x_{7-9} and x_{10-12} predicted in the second step.

sample, which benefits the model's generalization for the extension of diverse context lengths.

The extensible embeddings are learned with the downstream LLM's parameters (θ) fixed all the time. As a result, the LLM's original capabilities on short contexts are not affected by the introduction of extensible embeddings. Besides, we also empirically observe the strong but unexpected compatibility from the above training process, where the extensible embeddings can be directly applied to the fine-tuned derivatives of the downstream LLM without further adaptation.

2.4 Inference

The inference with the extensible embeddings is discussed w.r.t. the online and offline scenario, respectively. In particularly, the online scenario deals with the situation where the long-sequence data is streamingly presented (e.g., conversation). In this scenario, the generation process is conducted in consecutive sessions. In each session (i-th), the LLM predicts the new token ($t_{i,j}$) based on the extensible embeddings from the previous sessions ($Ex_{<i}$) and the preceding normal token embeddings within the current session ($\{x_{i,<j}\}$). The current session comes to its end when the total sum of both types of embeddings reaches the maximum capacity of context window (L^*): $|Ex_{<i}|+j = L^*$. Then, the normal token embeddings $\{x_{i,*}\}$ will be transformed into the extensible embeddings of the current session Ex_i and used by the next session.

The offline scenario handles the cases where the long-sequence data is fully presented beforehand (e.g., RAG, reading comprehension of long-docment). In this scenario, the extensible embeddings can be pre-computed for the data, which will significantly benefit the efficiency of online inference. In fact, it is OK to simply save the whole output embeddings from the extensible embedder, and flexibly sample for the extensible embeddings during inference based on the ad-hoc scaling factor.

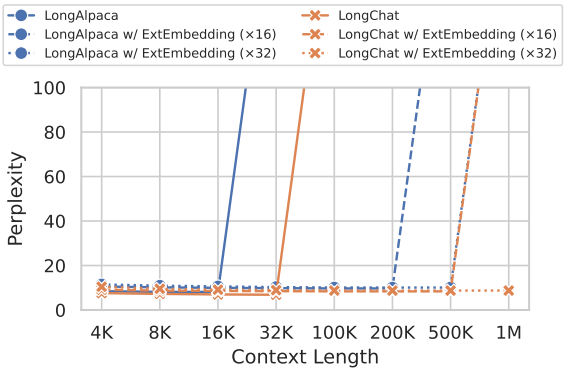

Figure 3: The extensible embedding trained on LLaMA-2-7B can be directly utilized by LongAlpaca-16K and LongChat-32K, leading to the scaling of their context lengths by $\times 16$ and $\times 32$ (PPL on PG19). Remarkably, the context of LongChat can be extended to 1 million.

3 Experiments

In this section, we conduct comprehensive experiments to investigate the following key issues about extensible embedding. 1) The effectiveness of context extension. 2) The flexibility and compatibility. 3) The running efficiency. 4) The technical factors about extensible embedding.

3.1 Experimental Settings

We leverage LLaMA-2-7B (chat) (Touvron et al., 2023) as our downstream LLM. We initialize extensible embedder with the first 8 layers of LLaMA-2-7B (chat). The training takes place on one Nvidia 8×A800 GPU machine, with a batch size of 8 and a learning rate of $5e^{-5}$ using the linear scheduler. The training is consecutively performed with 90K sampled instances from Redpajama (Computer, 2023) and 10K training instances from LongAlpaca (Chen et al., 2023b). Extensible embedding

Model	PG19					Books3				
	4K	**8K**	**16K**	**32K**	**100K**	**4K**	**8K**	**16K**	**32K**	**100K**
LLaMA-2-7B	7.77	$>10^3$	$>10^3$	$>10^3$	OOM	4.21	$>10^3$	$>10^3$	$>10^3$	OOM
PI	7.77	8.68	18.65	$>10^2$	OOM	4.21	5.99	11.4	69.8	OOM
NTK	7.77	8.13	10.71	55.22	OOM	4.21	5.10	7.71	52.3	OOM
StreamingLLM	7.98	8.01	8.00	8.00	8.00	4.32	4.34	4.33	4.33	4.34
LongAlpaca-16K	8.45	8.15	8.12	$>10^3$	OOM	4.93	4.67	4.64	$>10^3$	OOM
LongChat-32K	7.59	7.25	7.00	6.85	OOM	4.12	3.95	3.87	3.85	OOM
AutoCompressor-6K	26.9	$>10^3$	10^3	$>10^4$	OOM	17.1	$>10^3$	$>10^3$	$>10^4$	OOM
LongLLaMA	7.12	6.95	6.78	OOM	OOM	3.99	3.90	3.84	OOM	OOM
ExtEmbedding ($\times$16)	7.75	7.48	7.38	7.31	$>10^2$	4.32	4.20	4.15	4.13	$>10^3$
ExtEmbedding ($\times$32)	8.61	8.15	7.87	7.69	7.54	4.67	4.48	4.36	4.28	4.25

Table 1: Language modeling performance (measured by perplexity) on PG19 and Books3.

is trained with the downstream LLM's parameters always fixed. We consider the following baselines. The fine-tuning free methods: Positional Interpolation (PI) (Chen et al., 2023a), NTK-Aware Scaled RoPE (NTK) (ntk, 2023), StreamingLLM (Xiao et al., 2023b). The fine-tuned full-attention methods: LongAlpaca-7B-16K (Chen et al., 2023b), LongChat-7B-32K (Dacheng et al., 2023). The fine-tuned methods with modified architectures for long context: AutoCompressor-7B-6K (Rae et al., 2019), LongLLaMA-7B (Tworkowski et al., 2023). All baselines are based on LLaMA-2-7B, except LongLLaMA which leverages CodeLLaMA (Roziere et al., 2023).

3.2 Language Modeling

The long-context language modeling is evaluated with PG19 (Rae et al., 2019) and Books3 (Gao et al., 2020). Following the method used by Alexis et al. (Chevalier et al., 2023a), the perplexity is measured by predicting the last 512 tokens based on the preceding context. There are two evaluation settings about the extensible embedding: ExtEmbedding ($\times$16) and ExtEmbedding ($\times$32), where the scaling factor k is set as 16 and 32, respectively. The evaluation results are shown in Table 1, where the following observations can be derived.

On the one hand, we can observe the superior long-context language modeling quality achieved by extensible embedding. Firstly, extensible embedding leads to a notable improvement over the LLaMA-2-7B baseline, which indicates that the extended context can be effectively utilized to improve the generation quality. Secondly, the relative improvement (over LLaMA-2-7B) from extensible embedding is more pronounced than the fine-tuning free method. Although the fine-tuned full-attention methods may produce better performances in some cases, they require the change of the LLM's original parameters, and work with much higher running costs. Thirdly, extensible embedding is able to flexibly support much longer contexts by simply adjusting the scaling factor (k). In particular, by increasing the scaling factor from 16 to 32, LLaMA-2-7B's context length can be continually extended beyond $100K$ (up to $32 \times 4K$). The above observations validate the effectiveness of context extension with extensible embedding.

On the other hand, we can also make interesting observations about the extensible embedding's compatibility with the fine-tuned derivatives of its downstream LLM. We utilize two baseline models for evaluation: LongAlpaca and LongChat. Both models are fine-tuned from LLaMA-2-7B with long-sequence data, which achieve longer context windows of $16K$ and $32K$, respectively. As we can observe from Figure 3, the well-trained extensible embeddings on LLaMA-2-7B can be directly applied to the two models without any adaptation, which scales up their contexts by 16$\times$ and 32$\times$ times (with ExtEmbedding 16$\times$ and 32$\times$, respectively). Remarkably, we can reach a context length of 1 million tokens by enhancing LongChat-$32K$ with ExtEmbedding (32$\times$). We make further exploration with more fine-tuned derivatives (in Appendix B) and different evaluation tasks (§3.3), whose results affirm the ubiquity of this property.

3.3 Language Understanding

We evaluate long-context language understanding using 9 datasets from LongBench (Bai et al., 2023), which are about single-doc QA, multi-doc QA, and summarization. It's worth noting that the sequence lengths for majority of the evaluation samples are less than 16K or 32K. Therefore, the performance from the two fine-tuned methods LongAlpaca and LongChat can almost be a upper-bound for the rest of the methods. For each evaluation sample, the

Model	Len.	Single-Doc QA				Multi-Doc QA				Summarization			
		NQA	QASP	MF	Avg.	HQA	2WIKI	MSQ	Avg.	GOV	QS	MN	Avg.
LLaMA-2-7B	4K	18.70	19.20	36.80	24.90	25.40	32.80	9.40	22.60	27.30	20.80	25.80	24.70
PI	16K	12.85	20.86	23.24	18.98	22.41	22.13	6.94	17.16	29.77	18.48	26.85	25.03
NTK	16K	15.96	19.26	34.41	23.21	29.16	29.06	11.79	23.34	29.56	17.44	26.19	24.40
StreamingLLM	4K	18.95	16.84	26.16	21.47	30.09	26.87	11.03	22.22	25.81	20.64	25.88	22.20
LM. w. ExtEmbedding*	4K	19.59	23.18	33.90	25.56	34.53	32.93	13.30	26.92	27.06	21.15	25.69	24.63
LongAlpaca-16K (4k)	4K	17.85	27.66	34.91	26.81	29.50	31.32	12.50	24.44	30.09	23.12	27.57	26.93
LongAlpaca-16K (8k)	8K	18.60	28.89	38.35	28.61	32.32	30.84	11.34	24.83	31.43	24.42	27.87	27.91
LongAlpaca-16K	16K	19.13	28.91	37.03	28.36	36.93	30.32	17.23	28.16	31.30	24.16	27.84	27.77
LA. w. ExtEmbedding*	4K	20.12	29.45	36.25	28.61	37.58	33.54	13.84	28.32	30.45	22.66	27.52	26.88
LongChat-32K (4k)	4K	15.30	27.81	41.30	28.14	28.33	25.00	12.30	21.88	31.67	21.66	26.44	26.59
LongChat-32K (8k)	8K	17.35	29.14	41.67	29.39	29.58	24.65	10.83	21.69	32.40	22.30	26.39	27.03
LongChat-32K (16k)	16K	20.82	29.19	42.53	30.85	31.78	25.04	13.16	23.33	31.28	22.65	26.44	26.79
LongChat-32K	32K	21.00	29.25	42.70	30.98	32.99	24.86	14.02	23.96	31.03	23.00	26.44	26.82
LC. w. ExtEmbedding*	4K	17.07	30.59	42.69	30.12	31.52	25.84	13.17	23.51	31.19	20.62	26.77	26.19

Table 2: The evaluation of long-context understanding with tasks from LongBench. "(*k)" indicates that the input data is truncated to *k for the corresponding model. (LM.: LLaMA-2-7B, LA.: LongAlpaca, LC.: LongChat)

scaling factor is adjusted case-by-case for extensible embedding, which will let the compressed input just fit into the 4K context window of LLaMA-2-7B. The evaluation results are presented in Table 2, where the following observations can be made.

Firstly, extensible embedding can substantially improve upon the LLaMA-2-7B baseline for both single-doc QA and multi-doc QA. By comparison, the fine-tuning free methods bring in almost no contribution or even negative effect to long-context understanding, despite their effectiveness in language modeling. Although the fine-tuned methods bring seemingly better results on both QA and summarization tasks, we find that the improvement is mainly from the LLM's improved performance with short contexts (due to fine-tuning) rather than the incorporation of longer contexts. Particularly, both LongAlpaca and LongChat already achieve sufficiently high performances with the basic 4k context. For summarization, the further extension of context length is of little benefit. To some extent, the summarization cannot properly reflect the long context capability as most of the useful information for summarization is presented in the beginning or the end of each document, which has been covered by the basic 4K context. In the sense of relative improvement purely from the extended context, the extensible embedding's effect is comparable with fine-tuning. Meanwhile, it preserves a higher efficient because it only takes a 4K context.

Secondly, extensible embedding remains compatible with the fine-tuned derivatives of LLaMA-2-7B in this scenario. We directly apply the well-trained extensible embedding for LongAlpaca and LongChat. As introduced, the sequence lengths for the majority of evaluation samples are less than 16K or 32K. For those cases, the extensible embeddings will not introduce extra context, but only compress their original context to 4K, which makes the inference process more efficient. Notably, the two model's strong performances with the full-scale contexts can be effectively preserved by LA./LC. w. ExtEmbedding, which indicates two interesting properties: 1) the extensible embedding presents an almost lossless compression of the context, 2) the well-trained extensible embedding for LLaMA-2-7B can be seamlessly transferred to LongAlpaca and LongChat.

3.4 Efficiency Analysis

We analyze the efficiency in terms of GPU memory usage and inference time. The experiment is on a single Nvidia A800-80G GPU. The performance is measured by taking the average value of 100 forward passes where the last 512 tokens are predicted based on the input context. We include the following baselines. LongChat based on full-attention, where FlashAttention-2 (Dao, 2023) is enabled for its acceleration. StreamingLLM based on stream processing, whose window size is set to 2048; it is exempted from time evaluation because its current stepwise implementation is too slow. The extensible embedding uses a scaling factor $k = 32$, where both working modes are evaluated: the online mode where data is streamingly presented and extensible embeddings are generated step-by-step; the offline mode where data is presented in advance and extensible embeddings are pre-computed. The following observations can be made from Table 3.

Model	GPU Memory (GB)					Inference Time (s)				
	4K	8K	16K	32K	100K	4K	8K	16K	32K	100K
LongChat-32K	18.12	23.68	34.79	57.03	OOM	0.32	0.65	1.43	3.32	OOM
StreamingLLM	15.11	15.11	15.11	15.11	15.11	-	-	-	-	-
LongLLaMA	17.73	21.40	33.41	OOM	OOM	0.60	1.44	3.30	OOM	OOM
ExtEmbedding (online)	20.33	21.59	21.59	21.59	21.59	0.28	0.49	0.86	1.57	3.43
ExtEmbedding (offline)	13.96	14.21	14.75	15.79	17.54	0.08	0.08	0.10	0.12	0.23

Table 3: Efficiency analysis in terms of GPU memory usage and inference time.

First of all, the online mode leads to a constant memory usage, while the offline mode results in an even smaller consumption. As introduced (§2.4), the memory usage of extensible embedding comes from two parts. The first part is the generation of extensible embeddings, where the stream processing is conducted with a 4K sliding window (the offline mode is free from this step, thus taking even less GPU memory). The second part is the final inference stage based on the extensible embeddings, where the input sequence has been substantially condensed and become much shorter than its original length. Because the two operations are consecutively conducted, their memory costs will not accumulate. The online mode deals with both parts, but the overall memory cost is dominated by the first part. The offline mode only needs to handle the second part, which results in an even smaller cost. Both modes are free from processing the entire long input simultaneously, which contributes to a very economic usage of GPU memory.

Secondly, extensible embedding exhibits a much smaller time cost compared with the baseline methods. For ExtEmbedding (online), the majority of its computation is spent on the generation of extensible embedding. Because of the stream processing, the growth of its inference time is linear to the sequence length. Besides, with the pre-computation of the extensible embeddings, the inference time can be dramatically reduced for the offline mode. Such a superior time efficiency will substantially benefit extensible embedding's application in scenarios like RAG and long-doc QA, where long-sequence data can be presented in advance.

3.5 Ablation Studies

We perform ablation studies to analyze the influential factors about extensible embedding, whose results are presented with Table 4, 5, and 6.

Firstly, we explore the impact of down-scaling by comparing the default strided down-sampling (§2.2) with: 1) random down-sampling, which randomly chooses L/k output embeddings from the embedder, 2) terminal down-sampling, which selects the last L/k output embeddings from the embedder (L: chunk size). On one hand, the default strided down-sampling method outperforms the two baselines probably due to its more effective coverage of the context. On the other hand, the baselines can be directly applied to the embedder trained with strided down-sampling, which reflects the flexibility of usage of extensible embedding.

Factor	Setting	PG19	QA
Down scaling	Random down-sampling	7.64	23.39
	Terminal down-sampling	7.58	24.04
	Strided down-sampling*	7.31	25.56
Embedder size	First 4-layer (Llama-2-7B)	7.46	23.32
	First 8-layer (Llama-2-7B)*	7.31	25.56
Scale sampling	Monotonous ($k = 16$)	7.29	21.37
	Dynamic Sampling*	7.31	25.56

Table 4: Ablation studies. PG19 is measured by PPL under a 32K context; Single-Doc QA is measured by F1 score. Default settings are marked with "*".

Secondly, we analyze the impact from the size of embedder. Our default method uses the first 8 layers of LLaMA-2-7B, while the baseline uses the first 4 layers. It can be observed that the improved size leads to a better performance. There is no surprise about this observation because a larger embedder is more expressive and able to produce a better representation of the context. Nevertheless, it also comes with a larger computation cost. The optimal trade-off between cost and effectiveness must be determined for each scenario case-by-case.

Thirdly, we study the necessity of dynamically sampling the scaling factor during training (Scale sampling, §2.3). As a comparison, we employ a constant scaling factor k=16 (Monotonous). The Monotonous baseline achieves a comparable PPL on PG19 as our default method, because the language modeling task only performs a constant scaling down of the context by a factor of k=16. However, the default method notably outperforms Monotonous on Single-Doc QA, which relies on diversified scaling factors to condense the input of different lengths for a 4K context window.

We further investigate the impact of our training method based on two-stream AR (§2.3), where two

Training method	PPL (PG19) at different steps		
	100	500	1,000
Text Continuation	32.41	11.96	11.89
Auto-Encoding	10.27	9.95	10.25
Two-Stream AR*	8.85	8.17	8.00

Table 5: Impact from different training methods.

Scaling factor	Context Length				
	4K	8K	16K	32K	100K
$k = 2$	6.80	11.76	$>10^2$	$>10^2$	$>10^2$
$k = 4$	7.05	6.89	23.35	$>10^2$	$>10^2$
$k = 8$	7.40	7.18	7.13	27.10	$>10^2$
$k = 16$	7.75	7.48	7.38	7.31	$>10^2$
$k = 32$	8.61	8.15	7.87	7.69	7.54

Table 6: Impact from different scaling factors.

common strategies are introduced as our baselines (Mu et al., 2023; Chevalier et al., 2023b; Ge et al., 2023): 1) auto-encoding, where the input data is encoded in the first place and then decoded from the encoding result; 2) text continuation, where the head of input data (the first half) is encoded and the remaining part of the data is decoded from the encoding result. The experiment results are shown in Table 5, where two-stream AR notably outperforms the baselines. In just 100 steps, two-stream AR is able to achieve a remarkable performance of language modeling on PG19, which verifies it superior sample efficiency of training.

Finally, we make ad-hoc selection of scaling factor (k) and benchmark the PPL on PG19 at different context lengths (Table 6). On one hand, a smaller scaling factor, which means less compression of the data, can preserve a better generation quality. On the other hand, a larger scaling factor, which means higher compression of the data, will achieve a longer extension of the context. Consequently, the scaling factor should be properly selected in practice, such that the needed context can be fully covered with the lowest level of data compression.

4 Related Works

The extension of LLM's context is a critical issue. Recently, numerous methods have been proposed to tackle this problem from different perspectives. The popular approaches involve the modification of position encoding, e.g., Position Interpolation (Chen et al., 2023a) and NTK-Aware (ntk, 2023), which allows the LLMs to work with new positions during the inference time. The context extension quality from the modified position encoding can be improved by fine-tuning over long-sequence data (Peng et al., 2023). However, the fine-tuning is expensive, even with accelerations like LoRA (Chen et al., 2023b; Hu et al., 2021) and sparse attention (Chen et al., 2023b; Child et al., 2019). Besides, the fine-tuning operation may also bring unfavorable effect to the LLM's existing capability.

In addition to the increasing of window size, people also explore different methods to process a long context with a short context window. One common strategy is to leverage sliding windows (Chen et al., 2023a; Han et al., 2023), where the long context can be streamingly processed. However, the typical stream processing will simply ignore the information beyond the context window instead of making use of it. Another line of research is about context compression, which follows the same spirit as our method. In general, the context can be compressed in two optional ways. One is to explicitly compress the input data with methods like, summarization (Jiang et al., 2023), extraction (Jiang et al., 2023), or retrieval (Xu et al., 2023). Despite simplicity, the explicit compression is prone to incomplete and incoherent contextual information. The other option is to implicitly compress the input into latent embeddings (Bulatov et al., 2022; Mu et al., 2023; Chevalier et al., 2023b; Ge et al., 2023). The performance of implicit methods highly depend on the quality of compression, which is a joint result from the architecture of compressor and the learning method. So far, none of the previous methods are able to effectively realize a dramatic extension of LLM's context as extensible embedding (longer than 100K) due to the substantial loss of compression. The previous methods also lack the flexibility to support different context lengths. Besides, many of them need modifications on model architectures, which can be incompatible with the existing LLMs.

5 Conclusion

In this paper, we present extensible embedding as a new method to extend the LLM's context. It presents a compact representation for an extensible scope of context, which let the LLM to fully perceive the long-sequence input with its limited context window. It is realized based on a flexible model architecture and sample-efficient training algorithm, which not only optimizes the quality of context extension, but also leads to a remarkable flexibility and compatibility of usage, as well as a high efficiency of training and inference. The effectiveness of our method is verified with comprehensive evaluations, where the LLM's context can be dramatically extended with a superior quality.

References

2023. Localllama. ntk-aware scaled rope allows llama models to have extended (8k+) con- text size without any fine-tuning and minimal perplexity degradation. https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

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*.

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

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

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*.

Daixuan Cheng, Shaohan Huang, and Furu Wei. 2023. Adapting large language models via reading comprehension. *arXiv preprint arXiv:2309.09530*.

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

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023b. Adapting language models to compress contexts. In *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.

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

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

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

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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. *CoRR*, abs/2308.16137.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

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

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, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint*.

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

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

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

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

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

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

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

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

A The Overall Comparison of Extensible Embedding

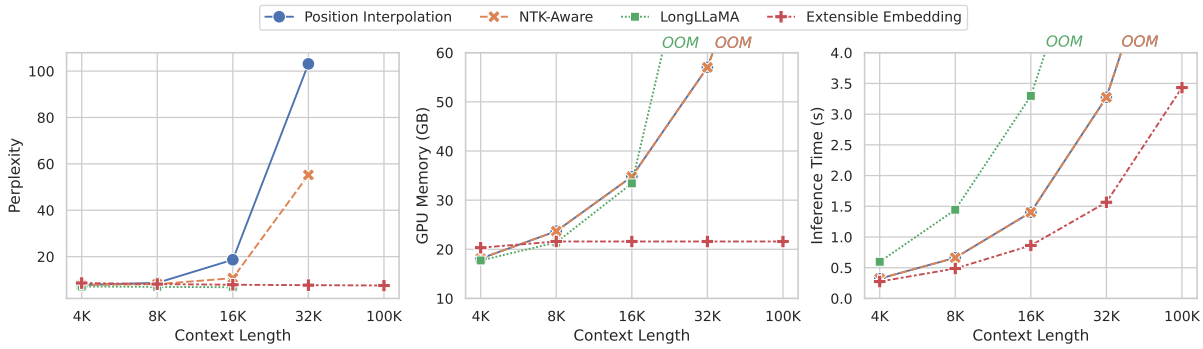

Figure 4: Comparison between extensible embedding and other context extension methods, including 1) Position Interpolation (Chen et al., 2023a), 2) NTK-Aware Scaled RoPE (ntk, 2023), 3) LongLLaMA (Tworkowski et al., 2023). Extensible Tokenization presents a superior long-context language modeling capability, along with better efficency in terms of memory and time. Perplexity is measured on PG19 (Rae et al., 2019) following the method in (Chevalier et al., 2023a)

B The Compatibility of Extensible Embedding

To further investigate the compatibility of extensible embedding, we directly apply the well-trained extensible tokenizer to more LLaMA-2-7B based model (Figure 5). We select the following models: 1) Vicuna-16K (Zheng et al., 2023), 2) Finance-Chat-4K (Cheng et al., 2023), 3) Law-Chat-4K (Cheng et al., 2023), and 4) Medicine-Chat-4K (Cheng et al., 2023). All these models are not only popular in the community, but have also been fine-tuned for specific domains. The method for measuring perplexity is consistent with main text.

Firstly, all these models extend their context length after implementing extensible embedding. For instance, with extensible embedding, Vicuna-16K's context length can extend up to 100K. Secondly, as the increase of the context length, the context generation quality improves. Thirdly, we find all these fine-tuned models perform well under different scaling factors, which indicates that extensible embedding maintains its flexibility when directly applying it to fine-tuned models.

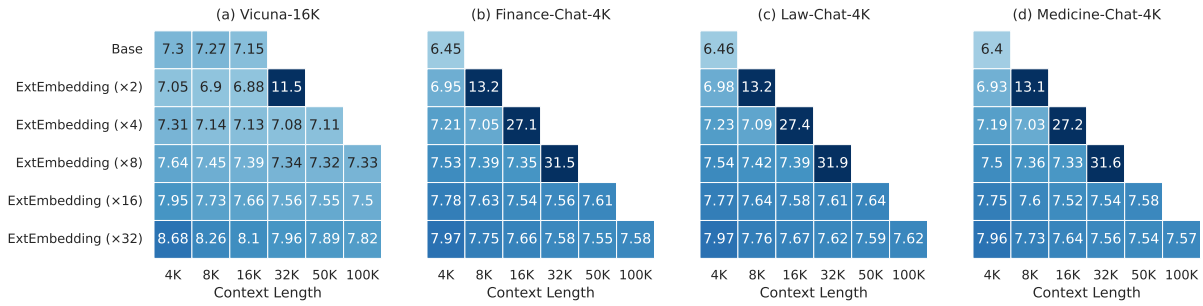

Figure 5: Compatibility of extensible embedding. The metric is perplexity on PG19. The darker the color, the higher the perplexity. And the blank area denotes instance where perplexity $> 10^2$.