LLM Maybe LongLM: SelfExtend LLM Context Window Without Tuning

Hongye Jin [1 *] **Xiaotian Han** [1 *] **Jingfeng Yang** [2] **Zhimeng Jiang** [1] **Zirui Liu** [3] **Chia-Yuan Chang** [1]
Huiyuan Chen [4] **Xia Hu** [3]

Abstract

It is well known that LLMs cannot generalize well to long contexts whose lengths are larger than the training sequence length. This poses challenges when employing LLMs for processing long input sequences during inference. In this work, we argue that LLMs themselves have inherent capabilities to handle long contexts without fine-tuning. To achieve this goal, we propose SelfExtend to extend the context window of LLMs by constructing bi-level attention information: the grouped attention and the neighbor attention. The grouped attention captures the dependencies among tokens that are far apart, while neighbor attention captures dependencies among adjacent tokens within a specified range. The two-level attentions are computed based on the original model's self-attention mechanism during inference. With minor code modification, our SelfExtend can effortlessly extend existing LLMs' context window without any fine-tuning. We conduct comprehensive experiments on multiple benchmarks and the results show that our SelfExtend can effectively extend existing LLMs' context window length. The code can be found at https://github.com/datamllab/LongLM.

1. Introduction

The context window length of most existing LLMs (Zhao et al., 2023; Yang et al., 2023) is limited since they are trained with a fixed length of training sequences. It's determined by the context window length during the pretraining stage. Once the length of the input texts exceeds the pretraining context window during the inference, the behavior of LLMs will be unpredictable and suffer from severe performance degradation. The perplexity (PPL) of the model will explode with the long input sequences (Xiao et al., 2023; Peng et al., 2023; Han et al., 2023; Chen et al., 2023b).

Recently, a variety of context window extension methods have been developed to extend the context window of pretrained LLMs. A straightforward approach is to fine-tune these models on enough extensive texts. Besides this, some methods seek to extend context window length in more efficient fine-tuning ways. Among these contemporary methods, some notable techniques include 'PI' (Chen et al., 2023b), 'CLEX' (Chen et al., 2023a) 'Yarn' (Peng et al., 2023), 'PoSE' (Zhu et al., 2023), 'LongLora' (Chen et al., 2023c), and 'ABF' (Xiong et al., 2023). These methods aim to extend the content window based on the implicit assumption that pretrained LLMs *lack the ability to handle long content*. However, these methods typically require finetuning to achieve extension, which can be resource and time-intensive given the quadratic complexity of Transformers. Additionally, high-quality long text data is scarce, hindering such fine-tuning approaches. Most real-world data is short, and much long text lacks meaningful long-range dependencies. With limited appropriate data, finetuning risks degrading existing strong performance on shorter sequences from pretraining or overfitting models to the tuning set. LLMs' generalizability to broad tasks may be reduced.

Instead of extending the content window, in this paper, we believe **LLMs should have inherent capabilities to handle long contexts**. Our belief stems from the fact that when we, as human beings, are children, we are taught how to read and write using relatively short texts, such as articles spanning several pages. We rarely use extremely long texts like entire books or complete documents as learning materials. Yet, we are still able to understand long texts effectively. With this strong motivation, the poor performance of LLMs while facing long text is not due to the lack of long context understanding capabilities. In our analysis, the key challenge preventing LLMs from effectively handling longer contexts is the Out-of-Distribution (O.O.D) issues related to positional encoding, which we call the *positional O.O.D*[1] issue. This

[*]Equal contribution [1]Texas A&M University [2]Amazon, the views expressed or the conclusions reached are his own and do not represent the view of Amazon [3]Rice University [4]Case Western Reserve University. Correspondence to: Hongye Jin <jhy0410@tamu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Here, the position refers to relative position rather than absolute position. The relative position is $m - n$ in RoPE, where m and n are the absolute positions of two tokens. The *positional*

problem arises when LLMs encounter text sequences during inference exceeding the length of their pretraining context window, where LLMs are exposed to new relative distances that were not present during their pretraining phase. It is widely recognized that Neural Networks (NNs) are susceptible to unpredictable behaviors when dealing with O.O.D inputs (Liu et al., 2021; Shen et al., 2021; Bai et al., 2021; Zhang et al., 2023). To address this, an intuitive and practical solution would be to remap the unseen relative positions to those encountered during the pretraining, thus extending the LLMs' ability to handle longer contexts naturally.

This paper proposes SelfExtend to elicit LLMs' inherent long context capabilities. SelfExtend addresses the issue of O.O.D. positional information by using a simple floor division operation to map unseen large relative positions to those encountered during pretraining. The core idea hinges on the observation that, in long texts, exacting word positions becomes less crucial. The overall meaning and the relative order of information hold greater significance. Just like when answering questions about lengthy texts, we rely on the general location and order, not the specific word-by-word placement. Natural language exhibits a characteristic where meaning stays relatively consistent within short ranges like paragraphs. Therefore, using close or even identical position encodings effectively captures the necessary relative ordering of important information. This intuitive approach aligns perfectly with the floor operation's functionality. Additionally, T5 (Raffel et al., 2020) and iRPE (Wu et al., 2021) also share this similar intuition.

Our SelfExtend is a plug-and-play method that takes effect at the inference stage, allowing existing large language models to easily adopt it. We evaluate SelfExtend with some popular LLMs (Llama-2 (Touvron et al., 2023), Mistral (Jiang et al., 2023a), SOLAR (Kim et al., 2023), and Phi-2 (Javaheripi et al., 2023)) on three types of tasks: language modeling, synthetic long context tasks, and real-world long context tasks. The proposed SelfExtend substantially improves the long context understanding ability and even outperforms many finetuning-based methods on some tasks. These results underscore SelfExtend as an effective solution for context window extension. The superior performance of SelfExtend also demonstrated the potential of large language models to effectively handle long contexts. Our main contributions are summarized as follows:

- We think LLMs with RoPE have a natural ability to handle long texts, even if they have not encountered super-long ones during training. The previous limitation stems from O.O.D. positions, meaning the "larger" positions have not been seen during training. We call this the *positional O.O.D.* issue.
- Based on this belief and to address the positional O.O.D. issue, we propose SelfExtend to extend the context window of LLMs without any fine-tuning. We map the unseen large relative positions (at inference) to known positions (at training), thus allowing LLMs to maintain coherence over longer texts without additional fine-tuning.
- In both synthetic and real-world long context tasks, SelfExtend has proven its ability to deliver performance that matches or surprisingly surpasses many existing fine-tuning-based models. This highlights the superior capabilities of our SelfExtend model.

O.O.D refers to cases where the value of $m - n$ during inference is unseen, i.e., larger than the values observed during pretraining. In this paper, we map unseen large relative positions to those observed during pretraining. More details about $m - n$ are provided in Section 2.

2. Preliminary

In this section, we present the preliminaries of our work.

Position Encoding. Transformers (Vaswani et al., 2017) incorporate position information via different positional embedding designs. The common positional embedding design can generally be categorized into two classes: absolute position embeddings and relative positional encodings. The *absolute position embedding* provides the absolute positions, which embeds each absolute position i into position vector $\mathbf{p}_i$ and adds word embeddings to their corresponding $\mathbf{p}_i$ before feeding them to the model. Examples of such include sinusoidal position embeddings (Vaswani et al., 2017) and learned position embeddings in GPT3 (Brown et al., 2020) and OPT (Zhang et al., 2022), or adding the dot product between two tokens' position embeddings on the attention logit (Ke et al., 2020). On the other hand, relative positional encodings have been proposed to use relative distance information between tokens and have become the mainstream of position embedding. This information is usually applied in attention layers. Examples of such include a learnable attention logit bias as in T5 (Xue et al., 2020), Transformer-XL (Dai et al., 2019); a fixed linear attention decay called Alibi (Press et al., 2021); rotating query and key sequences based on distance such as RoPE (Su et al., 2022), and XPos (Sun et al., 2023). The proposed method in this work is based on the Rotary Position Embedding (RoPE) introduced in (Su et al., 2022).

RoPE. Here, we introduce the basic concept of RoPE. Let's consider a sequence of tokens represented as $w_1, w_2, \cdots, w_L$, and their corresponding embeddings are denoted as $\mathbf{x}_1, \cdots, \mathbf{x}_L \in \mathbb{R}^{|D|}$, where $|D|$ is the dimension of the embedding. The basic idea of RoPE is to incorporate the positional information into the query $\mathbf{q}$ and the key vectors $\mathbf{k}$, respectively. This integration ensures that their inner product $\mathbf{q}^T\mathbf{k}$ will contain the relative positional embedding information inherently. To achieve this, RoPE employs the

following vector transformations:

$$\mathbf{q}_m = f_q(\mathbf{x}_m, m) \in \mathbb{R}^{|L|},\ \mathbf{k}_n = f_k(\mathbf{x}_n, n) \in \mathbb{R}^{|L|}, \quad (1)$$

where $|L|$ is the hidden dimension of per head. The functions f_q and f_k responsible for injecting positional information, are defined as $f_q(\mathbf{x}_m, m) = W_q\mathbf{x}_m e^{im\theta}$, $f_k(\mathbf{x}_n, n) = W_k\mathbf{x}_n e^{in\theta}$, where $\theta_d = b^{-2d/|D|}$, $b = 10000$ and projectors $W_q, W_k : \mathbb{R}^{|D|} \rightarrow \mathbb{R}^{|L|}$. RoPE keeps the real part of the inner product $\mathbf{q}^T\mathbf{k}$, which is $\text{Re}(\mathbf{q}^*\mathbf{k})$. This operation ensures that the dot product of the query and key vectors depends entirely on the relative distance between the tokens, represented by $m - n$ of the tokens as follows:

$$\langle f_q(\mathbf{x}_m, m), f_k(\mathbf{x}_n, n)\rangle_{\mathbb{R}} = \text{Re}(\langle f_q(\mathbf{x}_m, m), f_k(\mathbf{x}_n, n)\rangle_{\mathbb{C}})$$
$$= \text{Re}(\mathbf{x}_m^* W_q^* W_k \mathbf{x}_n e^{i\theta(m-n)}) = g(\mathbf{x}_m, \mathbf{x}_n, m-n), \quad (2)$$

where $g(\cdot)$ is an abstract mapping function.

3. SelfExtend

In this section, we first conduct a preliminary investigation on the inherent ability of the LLMs to handle long content. Then, we propose our SelfExtend that effectively extends existing LLMs' context window without any fine-tuning.

3.1. Preliminary Analysis

① Why do LLMs fail on sequences during inference that are longer than their pre-training context window? For a pretrained LLM with relative position encodings, such as RoPE, the behavior of the LLMs becomes unpredictable during inference if the length of a sequence is longer than its pretraining context window length. This has been explored by (Han et al., 2023; Chen et al., 2023b) that with unseen relative positions, the attention distributions are different compared to those within the pretraining context window. We argue that such failure stems from the Out-of-Distribution (O.O.D.) relative distance in the sense that neural networks are not robust to O.O.D. inputs (Shen et al., 2021).

② How to solve positional O.O.D. problem? One feasible and straightforward way to handle unseen relative positions is to map them to positions that were seen during pretraining. We can use the FLOOR operation to map the unseen positions to positions within the pretraining context window, as shown in Figure 1. The proposed method is identical to the original self-attention mechanism except that the FLOOR operation is applied to each token's original position before the inner product. We denote the self attention with the FLOOR operation applied as "grouped attention". In Python style, the "grouped attention" is denoted as:

$$P_g = P \ \ // \ \ G_s, \quad (3)$$

where $P \in \mathbb{R}^{B\times L}$ is the original position encoding, in which B is the batch size and L is the length of the input text sequence. G_s denotes the group size, which is the base of the FLOOR operation. Taking the floor of the position divided by the group size maps the original large position values to a smaller discrete set of values, avoiding the issue of out-of-distribution position values during inference.

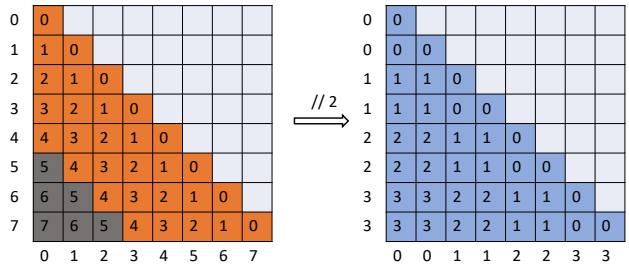

Figure 1. Illustration of grouped attention. We suppose that the LLM's pretraining context window length is 5 and the length of the inference sequence is 8. On the left figure, we show the positional Out-of-Distribution (O.O.D.) issue while the input length is out of the pretraining context window size. The y-axis of this matrix represents the position of query tokens and the x-axis represents the position of key tokens. In this case, in the relative position matrix, only those in orange are seen during pretraining. Relative positions in gray are outside the pretraining context window. In the right figure, we show how the FLOOR operation is applied and the relative position matrix for grouped self attention. With the G_s set as 2, the positions of query tokens and key tokens are mapped from 0-7 to 0-3 by FLOOR (//). The new relative positions (in blue) are all within the range of the pretraining context window.

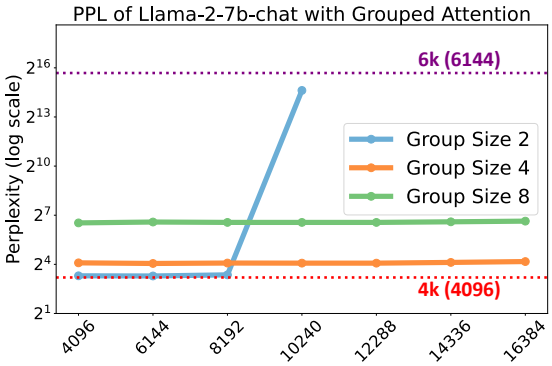

Figure 2. Perplexity (PPL) using grouped attention with different group sizes under different sequence lengths on PG-19 dataset. The original Llama-2-7b-chat PPL is stable at 4k (4096) sequences (red dotted line) but explodes at 6k (6144) sequences (purple dotted line). The results show the LLMs keep a relatively low and stable PPL on long sequences with grouped attention.

③ Can LLMs work well without accurate position information? — Yes, but not that perfect. We show the perplexity (PPL) on the PG-19 (Rae et al., 2019) dataset with the FLOOR operation applied to Llama-2-7b-chat across different sequence lengths, in Figure 2. As a comparison, we also show the PPL of the original model without the FLOOR operation as the dotted lines. From this figure, with the FLOOR operation, LLMs keep a relatively low and stable PPL on the sequences whose lengths exceed the pretraining context window. Meanwhile, with grouped attention, the

PPL is a little higher than the original LLMs, which is expected. However, the model's PPL behavior is similar to the original model, as the PPL is nearly unchanged within the "context window" (for Llama-2: 2 - 8192, 4 - 16384, and 8 - 32768), demonstrating the effectiveness of group attention. Once the length of a sequence is longer than the new "context window" (e.g., sequences with 10k tokens as the input, with a group size of 2), the PPL explodes again due to the *positional O.O.D* issue.

④ **How to restore degraded language modeling ability caused by grouped attention? — Re-introducing normal attention in the neighboring area.** In the process of generating next tokens, the immediate neighbors of a target token play a crucial role, which is well-supported by existing methods of sparse attention mechanisms (Zaheer et al., 2020; Shi et al., 2021) and methods for extending the contextual window (Han et al., 2023; Xiong et al., 2023; Liu et al., 2024). These studies consistently highlight the importance of maintaining the standard attention mechanism for tokens in close proximity to the target token. This proximity-based focus is essential for the accurate generation of the next token, ensuring the coherence and fluency of the generated text, as evidenced by acceptable perplexity (PPL) levels. Employing grouped attention might not significantly affect the overall quality of generated sentences; however, it necessitates the accurate positioning of attention to maintain generation quality. Therefore, it is imperative to preserve the standard attention mechanism within the vicinity of the target token, as utilized during the pretraining phase, to ensure the precision and effectiveness of language models in capturing the nuances of local context.

3.2. SelfExtend LLM Context Window Without Tuning

We introduce SelfExtend, a method that enhances LLMs' natural capability to process extensive contexts without the need for fine-tuning. SelfExtend incorporates two distinct types of attention mechanisms: 1) Grouped attention, specifically designed for tokens that are far apart. This approach applies a floor operation to the positions to manage long-distance relationships between tokens; 2) Standard attention, which employs the conventional attention mechanism for adjacent tokens within a specified range. The SelfExtend framework is depicted in Figure 3. Notably, SelfExtend modifies only the attention mechanism during inference, eliminating the need for additional fine-tuning.

Maximum Extended Length of SelfExtend Suppose that we have the pretraining context window size as L, the group size for grouped attention as G_s, and the window size for neighbor tokens as w_n. We shift the relative position of grouped attention by $w_n - w_n//G_s$ before merging the two pieces of attention together. This ensures that the transition from the normal attention area to the grouped attention area smooth. We merge the two parts of attention by replacing the attention values out of the neighbor token window with the attention values from the grouped attention. All the modifications are applied before the softmax operation and other parts remain unchanged. Ideally, the maximum length of the extended context window is:

$$(L - w_n) * G_s + w_n. \tag{4}$$

For example, in Figure 3, the context window is extended from its pretraining length of 7 to $(7-4)*2+4=10$. The pseudo code for SelfExtend is presented in Algorithm 1.

Table 1. Perplexity on dataset PG19 with Llama-2-7b-chat and Mistral-7b-instruct-0.1. We report the PPL of with&without Sliding Window Attention (SWA) for Mistral.

Model	Evaluation Context Window Size						
Name	**4096**	**6144**	**8192**	**10240**	**12288**	**14336**	**16384**
Llama-2-7b-chat	9.181	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
SelfExtend-Llama-2-7b-chat	8.885	8.828	9.220	8.956	9.217	9.413	9.274
Mistral-7b-instruct-0.1 w/ SWA	9.295	9.197	9.532	9.242	9.198	9.278	9.294
Mistral-7b-instruct-0.1 w/o SWA	9.295	9.205	10.20	55.35	$> 10^3$	$> 10^3$	$> 10^3$
SelfExtend-Mistral-7b-instruct-0.1	9.272	9.103	9.369	9.070	8.956	9.022	9.128

Relation to Existing Work The grouped attention in SelfExtend can be viewed as a form of position interpolation (Chen et al., 2023b), where some positions are interpolated to be infinitely close to pretraining positions. Another finetuning-free method, ReRoPE (Su, 2023), is equivalent to a special case of SelfExtend: the group size is large enough that all tokens outside the neighbor window fall into the same group (e.g. group size $10,000$ in Figure 5). T5 (Raffel et al., 2020) and iRPE (Wu et al., 2021) also share the high-level idea of multi-level positional encodings, while applying it during pretraining. T5 is more similar to ReRoPE for using the same position for distant tokens. iRPE has finer distant position encodings, more akin to SelfExtend.

4. Experiments

We evaluate SelfExtend with Llama-2 (Touvron et al., 2023) and its families, Phi-2 (Javaheripi et al., 2023), Mistral (Jiang et al., 2023a) and SOLAR (Kim et al., 2023) on language modeling task, synthetic long context tasks, real-world long context tasks and standard short-context tasks.

4.1. Performance on Language Modeling Tasks

Language modeling task is the most fundamental and the least requirement for LLMs, which is usually measured by perplexity (PPL) on the test text data. A low PPL does not guarantee good performance on real tasks (Pal et al., 2023), however, a higher PPL suggests severe performance degradation of LLMs. We evaluate SelfExtend's language modeling performance on dataset PG19 (Rae et al., 2019), which contains lengthy books. PPL is used as the metric. More experimental details are presented in Appendix D.1

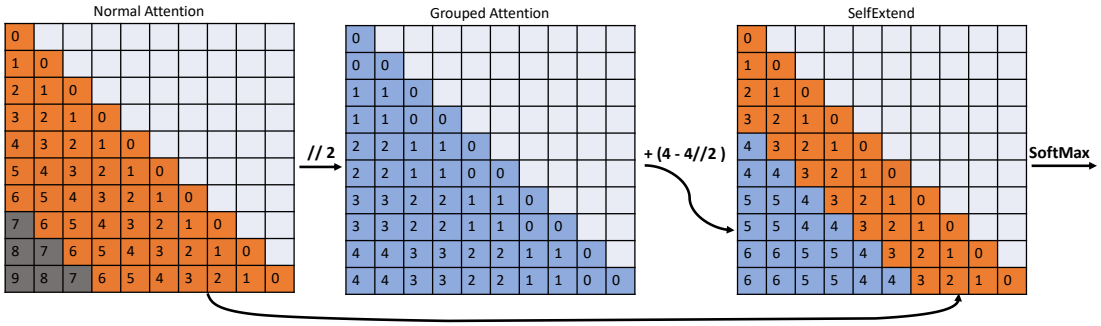

Figure 3. Illurstation of SelfExtend. This figure shows the attention score matrix (before SoftMax operation) of SelfExtend while a sequence of length 10 is fed into an LLM with the pretraining context window size ($L = 7$). The numbers denote the relative distances between the corresponding query and key tokens. SelfExtend has two kinds of attention mechanism: for neighbor tokens within the neighbor window ($w_n = 4$), it adapts the normal self-attention; for tokens out of the window, it adapts the values from the grouped attention. The group size (G_s) is set to 2. We then merge two parts attention matrices and apply the softmax operation.

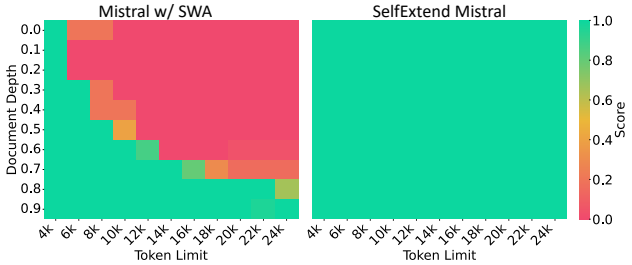

Figure 4. Passkey retrieval accuracy for Mistral-7b-instruct-0.1 with SWA or SelfExtend. Mistral with SelfExtend obtains 100% passkey retrieval accuracy For all sequence length (token limit) and all depth. Mistral with SWA cannot retrieve the passkey out of the sliding window. The default sliding window size is 4096.

The results show that SelfExtend can successfully maintain a low PPL out of the pretraining context window for both Llama-2-7b-chat and Mistral. Without SelfExtend, the PPL explodes when the length of test sequence is larger than the context window. Mistral with SWA can also maintain a low PPL out of its context window. But later in the next section, we will demonstrate that a low PPL score does not necessarily indicate proficiency in handling long contexts. More discussion about PPL can be found in Appendix B.

4.2. Performance on Synthetic Long Context Tasks

The passkey retrieval task is the same as what is defined in Landmark Attention (Mohtashami & Jaggi, 2023), which is a synthetic long context task. It requires a language model to retrieve a simple passkey (i.e., a 5-digit random number) in a long meaningless text sequence. The passkey is placed with various document depths (where the passkey is placed in the input texts) and context lengths (ranging from 4k to 24k). We tested multiple passkey retrievals for each context length and depth. The passkey was randomly placed within a span of 400 tokens. For a depth of 0.1 and context of 8k, the passkey was placed between tokens $800 - 1600$. We performed 10 iterations per span, so 20 total for that setting. Experimental setting details and an example of passkey retrieval task can be found in Appendix D.2.

The results in Figure 4 show that without any fine-tuning, SelfExtend obtains **100%** passkey retrieval accuracy across all tested depths and context lengths. The results also demonstrate that: although Mistral w/ SWA has low PPL beyond its pretraining context window, it can only access information (i.e. the passkey) within its sliding window. Considering the simplicity of this task, these results strongly suggest it still does not have the true ability to handle long contexts.

4.3. Performance on Real-World Long Context Tasks

Evaluation solely on language modeling (measured by perplexity) and synthetic tasks like passkey retrieval cannot fully assess the long-context capabilities of LLMs. The task of Passkey retrieval is overly straightforward, and an LLM may still struggle with long context despite low perplexity. To comprehensively evaluate long-context performance, we further use two recent real-world long context benchmarks: LongBench (Bai et al., 2023) and L-Eval (An et al., 2023). The results are presented in Table 2 and Table 3. On the LongBench in Table 2, for all four different base LLMs and most datasets, with SelfExtend, the LLM can obtain significant performance improvements.

Llama-2-7B: We use SelfExtend to increase Llama-2-7b-chat's context from 4k to 16k and 25k. Both significantly outperform Llama-2-7b-chat and most fine-tuned models on several datasets like HotpotQA. We also extend vicuna1.5-7B from 4k to 16k and 25k. With SelfExtend, vicuna1.5-7B surpasses its fine-tuned counterpart vicuna1.5-7B-16k and ranks among top Llama-2-7b models. On some datasets, the 25k variant underperforms the 16k one due to the trade-off between larger context and positional precision. More details about the trade-off is in Section 4.5.

Mistral-7B: We extend Mistral-7B's context to 16k, significantly improving its long context ability over the base model. The fine-tuned variant MistralLite ((amazon, 2023))

Table 2. Performance comparison of different LLMs on LongBench. * indicates the results reported by LongBench. *indicates the results are reported by CLEX (Chen et al., 2023a). + indicates the results from us. Models in green/blue/cyan/orange are based on Llama2-7b/Mistral-7b/Phi-2/SOLAR-10.5B. The number (e.g. '25k') indicates the maximum input length. The 'SE' prefix indicates SelfExtend is applied to this model. In this table, except SelfExtend, all other models require fine-tuning to extend the context window. CLEX is fine-tuned with 2B tokens. LongChat1.5-7B-32k and Vicuna1.5-7B-16K are fine-tuned on more than 80k conversations. CodeLLaMA (Rozière et al., 2023) is fine-tuned on more than 500B tokens. MistralLite (Yin Song and Chen Wu and Eden Duthie, 2023) is also fine-tuned on more than 2B tokens (amazon, 2023). The better performance between models w/ and w/o SelfExtend is in **bold**.

	LLMs[a]	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code	
		NarrativeQA	Qasper	MultiField-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PassageCount	PassageRe	Lcc	RepoBench-P
SelfExtend	Llama-2-7B-chat-4k*	18.7	19.2	**36.8**	25.4	32.8	9.4	27.3	20.8	25.8	61.5	77.8	40.7	2.1	**9.8**	52.4	43.8
	SE-Llama-2-7B-chat-16k+	**21.69**	25.02	35.21	34.34	30.24	14.13	27.32	**21.35**	25.78	**69.50**	**81.99**	40.96	**5.66**	5.83	**60.60**	**54.33**
	SE-Llama-2-7B-chat-25k+	21.37	**26.68**	34.63	**35.47**	**30.46**	**15.51**	**27.51**	21.30	**25.87**	68.50	78.79	**41.29**	3.90	3.50	59.69	53.83
	Mistral-7B-ins-0.1-16k w/ SWA+	19.40	34.53	37.06	42.29	32.49	14.87	27.38	22.75	26.82	65.00	**87.77**	42.34	**1.41**	28.50	**57.28**	53.44
	Mistral-7B-ins-0.1-8k w/o SWA+	20.46	35.36	39.39	34.81	29.91	11.21	24.70	21.67	26.67	68.00	86.66	41.28	0.18	24.00	56.94	**55.85**
	SE-Mistral-7B-ins-0.1-16k+[b]	**23.56**	**39.33**	**49.50**	**45.28**	**34.92**	**23.14**	**30.71**	**24.87**	**26.83**	**69.50**	86.47	**44.28**	1.18	**29.50**	55.32	53.44
	Phi-2-2k+	4.46	7.01	19.98	**9.43**	8.55	**4.62**	25.64	14.32	**24.03**	50.50	74.55	**1.71**	**2.83**	**4.17**	**58.96**	54.14
	SE-Phi-2-8k+	**12.04**	**12.10**	**20.15**	8.22	**9.68**	3.89	**27.90**	**14.58**	22.13	**61.00**	**82.82**	1.40	2.37	2.83	57.87	**56.42**
	SOLAR-10.7B-ins-4k+	16.50	24.06	46.76	44.03	**36.05**	22.76	**31.39**	19.81	**26.36**	70.00	87.91	42.49	**4.5**	26.5	41.04	54.36
	SE-SOLAR-10.7B-ins-16k+	**22.63**	**32.49**	**47.88**	**46.19**	34.32	**27.88**	30.75	**22.10**	25.62	**74.50**	**89.04**	**42.79**	4.0	**28.0**	**53.73**	**56.47**
Other Methods	LongChat1.5-7B-32k*	16.9	27.7	41.4	31.5	20.6	9.7	30.8	22.7	26.4	63.5	82.3	34.2	1.0	30.5	53.0	55.3
	together/llama-2-7b-32k+	15.65	10.49	33.43	12.36	12.53	6.19	29.28	17.18	22.12	71.0	87.79	43.78	1.0	23.0	63.79	61.77
	CLEX-7B-16k*	18.05	23.68	44.62	28.44	19.53	9.15	32.52	22.9	25.55	68	84.92	42.82	0	11.5	59.01	56.87
	CodeLLaMA-7B-16k*	22.93	30.69	43.37	33.05	27.93	14.2	28.43	24.18	26.84	70	84.97	43.43	2	13.5	64.35	55.87
	SE-Llama-2-7B-chat-16k+	21.69	25.02	35.21	34.34	30.24	14.13	27.32	21.35	25.78	69.50	81.99	40.96	5.66	5.83	60.60	54.33
	SE-Llama-2-7B-chat-25k+	21.37	26.68	34.63	35.47	30.46	15.51	27.51	21.30	25.87	68.50	78.79	41.29	3.90	3.50	59.69	53.83
	Vicuna1.5-7B-16k*	19.4	26.1	38.5	25.3	20.8	9.8	27.9	22.8	27.2	71.5	86.2	40.8	6.5	4.5	51.0	43.5
	SE-Vicuna1.5-7B-16k+	21.88	35.16	42.00	31.14	22.51	13.33	28.47	22.24	26.70	69.50	86.31	40.54	3.56	7.50	60.16	44.07
	SE-Vicuna1.5-7B-25k+	22.46	34.42	42.58	30.95	24.33	12.72	27.75	22.26	27.21	72.00	84.02	40.38	3.01	7.00	58.86	43.86
	MistralLite-16k+	32.12	47.02	44.95	58.5	47.24	31.32	33.22	26.8	24.58	71.5	90.63	37.36	3	54.5	66.27	65.29
	SE-Mistral-7B-ins-0.1-16k+	23.85	37.75	46.93	45.35	34.54	23.28	30.45	23.58	26.94	69.50	85.72	43.88	0.59	28.50	54.92	53.44
Fixed Models	GPT-3.5-Turbo-16k*	23.6	43.3	52.3	51.6	37.7	26.9	29.5	23.4	26.7	68.0	91.4	41.7	4.5	71.0	54.7	53.6
	XGen-7B-8k*	18	18.1	37.7	29.7	21.1	10.3	27.3	20.5	26.2	65.5	77.8	25.3	2.1	8.5	38.6	38.6
	InternLM-7B-8k*	12.1	16.7	23.4	28.7	22.8	9.0	9.7	15.9	22.8	52.0	77.8	21.2	3.0	6.0	44.1	28.8
	ChatGLM2-6B-32k*	21.1	31.5	46.2	45.1	34.0	21.9	32.4	24.0	26.5	62.5	78.7	36.3	1.5	77.0	55.6	49.9
	ChatGLM3-6B-32k*	26.0	43.3	51.7	54.4	44.9	40.4	36.8	23.9	27.9	79.0	87.1	38.2	2.0	99.0	57.66	54.76
	Baichuan-13B-4k*	0.07	17.55	17.28	3.29	15	0.1	6.8	1.71	23.1	20.05	20.06	5.77	0.06	0.5	47.98	16.58
	ALiBi-7B-4k*	0.04	8.13	17.87	2.73	8	1.33	5.31	1.64	25.55	9.25	8.83	4.67	0	1.27	46.69	18.54

[a] Details of used LLMs in this table are presented in Appendix E.

Table 3. Exam evaluation results on L-Eval. **Tokens** denotes the maximum input context length. + indicates the results are from us and others are reported by L-Eval. The rows in the same color (orange, green, blue, and pink) represent the models of those rows from the same base model. The better performance between models w/ and w/o SelfExtend is highlighted in **bold**.

Model	Tokens	Coursera	GSM	QuALITY	TOEFL	CodeU	SFiction	Avg.
Claude1.3-100k	100k	60.03	88.00	73.76	83.64	17.77	72.65	65.97
GPT-4-32k	32k	75.58	96.00	82.17	84.38	25.55	74.99	73.11
Turbo-16k-0613	16k	63.51	84.00	61.38	78.43	12.22	64.84	60.73
Chatglm2-6b-8k	2k	43.75	13.00	40.59	53.90	2.22	54.68	34.69
XGen-7b-8k (2k-4k-8k)	2k	26.59	3.00	35.15	44.23	1.11	48.43	26.41
Chatglm2-6b-8k	8k	42.15	18.00	44.05	54.64	2.22	54.68	35.95
Chatglm2-6b-32k	32k	47.81	27.00	45.04	55.01	2.22	57.02	39.01
XGen-7b-8k	8k	29.06	16.00	33.66	42.37	3.33	41.40	27.63
MPT-7b-65k	8k	25.23	8.00	25.24	17.84	0.00	39.06	19.22
Llama2-7b-chat	4k	29.21	19.00	37.62	51.67	1.11	60.15	33.12
Longchat1.5-7b-32k	32k	32.99	18.00	37.62	39.77	**3.33**	57.02	31.45
Llama2-7b-NTK	16k	32.71	19.00	33.16	52.78	0.00	**64.84**	33.74
SE-Llama2-7B-chat+	16k	**35.76**	**25.00**	**41.09**	**55.39**	1.11	57.81	**36.02**
Vicuna1.5-7b-16k	16k	**38.66**	19.00	39.60	**55.39**	**5.55**	60.15	36.39
SE-Vicuna1.5-7B+	16k	37.21	**21.00**	**41.58**	**55.39**	3.33	**63.28**	**36.96**
Llama2-13b-chat	4k	35.75	39.00	**42.57**	60.96	1.11	54.68	39.01
Llama2-13b-NTK	16k	36.48	11.00	35.64	54.64	1.11	**63.28**	33.69
Llama2-13b-NTK(Dyn)	16k	30.08	**43.00**	41.58	64.31	1.11	35.15	35.87
SE-Llama2-13B-chat+	16k	**38.95**	42.00	41.09	**66.17**	1.11	**63.28**	**42.10**
Mistral-7b-ins-0.1 w/ SWA+	16k	44.77	44.00	46.53	60.59	2.22	**64.06**	43.70
Mistral-7b-ins-0.1 w/o SWA+	8k	43.60	49.00	45.05	60.59	**4.44**	60.94	43.94
MistralLite+	16k	29.23	32.00	46.04	17.47	3.33	14.06	23.69
SE-Mistral-7b-ins-0.1+	16k	**45.20**	**51.00**	**48.02**	**64.68**	3.33	59.38	**45.27**
Phi-2+	2k	38.37	64.00	**42.08**	55.76	3.33	**52.34**	42.64
SE-Phi-2+	8k	**42.44**	**65.00**	41.08	**62.83**	**4.44**	**52.34**	**44.69**
SOLAR-10.7b-Instruct-v1.0+	4k	48.84	**72.00**	59.90	77.32	**4.44**	69.53	55.34
SE-SOLAR-10.7b-v1.0+	16k	**50.44**	**72.00**	**70.30**	**79.18**	**4.44**	**73.44**	**58.30**

achieves the best performance on most datasets. However, many of these datasets were included in MistralLite's fine-tuning data, such as NarrativeQA[2].

SOLAR-10.7B and Phi-2: They have no finetuned variant for context window extension yet. SelfExtend can also obtain substantial performance improvements.

On the LEval benchmark in Table 3, we observe similar results. Compared to fine-tuning free baselines like NTK or further fine-tuned models like Longchat1.5-7b-32k and Vicuna1.5-7b-32k, SelfExtend achieves superior performance on nearly all datasets[3].

In summary, on the two benchmarks, **SelfExtend achieves comparable or better performance, compared to methods that requires further fine-tuning**. Despite our initial expectation being that SelfExtend would simply outperform

[2]More details about MistralLite's fine-tuning data can be found at `https://huggingface.co/amazon/MistralLite`. At least, GovReport, QMSum, NarrativeQA, Qasper, QuALITY, and HotpotQA are included. Meanwhile, Multi-passage QA and summarization tasks are also in fine-tuning data. This also violates zero-shot evaluation conditions.

[3]LEval performance seems sensitive to prompt engineering for these sub-13B LLMs. For example, on some datasets, vanilla vicuna-13b underperforms vanilla vicuna-7b.

Table 4. Performance of SelfExtend on Hugging Face Open LLM benchmark compared to baselines: Llama 2, Llama-2-chat-4, Mistral-instruct-v0.1 and Phi-2. We use the same hyper-parameters as on LongBench benchmark. For Llama-2 & Llama-2-chat based SelfExtend, the group size is 16 and neighbor window is 1024; for Mistral based SelfExtend, the group size is 6 and neighbor window is 1024; for Phi-2 based SelfExtend, the group size is 12 and neighbor window is 512.

Size	Name	ARC-c	Hellaswag	MMLU	TruthfulQA	GSM8k
7B	Llama-2	**52.99**	**78.66**	46.58	**38.97**	**14.94**
7B	SE-Llama 2	**52.99**	78.65	**46.68**	**38.97**	14.71
7B	Llama-2-chat	**52.73**	**78.49**	**48.20**	45.32	18.73
7B	SE-Llama-2-chat-16k	**52.73**	**78.49**	48.09	**45.33**	**18.88**
7B	Mistral-instruct-v0.1	54.35	**75.72**	55.57	**55.89**	30.93
7B	SE-Mistral-instruct-v0.1	**54.44**	75.71	**55.59**	**55.89**	**31.39**
2.7B	Phi-2	**61.17**	75.13	58.20	**44.54**	55.11
2.7B	SE-Phi-2	61.00	**75.20**	**58.29**	**44.54**	**55.42**

the base model without additional extension methods, it is remarkable that our SelfExtend, which solely operates during inference without the need for fine-tuning or training, achieves such impressive performance.

4.4. Performance on Short Context Tasks

We argue that an ideal context length extension method should not degrade performance on standard short-context tasks. Previous fine-tuning based methods usually undergo performance degradation on short-context tasks (Peng et al., 2023; Xiong et al., 2023). Following (Peng et al., 2023), we use Hugging Face Open LLM Leaderboard (Gao et al., 2023) to evaluate SelfExtend's performance on five public short context tasks. Specifically, we use 25-shot ARC-Challenge (Clark et al., 2018), 10-shot HellaSwag (Zellers et al., 2019), 5-shot MMLU (Hendrycks et al., 2020), 0-shot TruthfulQA (Lin et al., 2021), and 5-shot GSM8K (Cobbe et al., 2021). The results are shown in Table 4. We also investigate the influence of varying group sizes and neighbor window sizes on short-context tasks and we present the results in Appendix C.

The results show that SelfExtend can maintain the performance of the short-context tasks, while enhance the performance on long-context tasks. Moreover, because SeldExtend does not require any fine-tuning and only takes effect during inference, SelfExtend can be readily adopted as a plug-in component for LLMs. This means SelfExtend can be automatically and inherently disabled while encountering short-text sequences. Then, with the parameters remaining unchanged, LLMs can maintain its original inference mechanism on those short-context scenarios.

4.5. Ablations on Group Size and Neighbor Window

We investigate the influence of varying the group size G_s and the neighbor window w_n. We experiments with Phi-2 on four real-world datasets from Longbench: narrativeqa, qasper, triviaqa, and repobench-p. The results are presented in Figure 5. Form the results, we observe two trade-offs:

Table 5. Performance of Phi-2 with different context window lengths. The vanilla Phi-2 has a 2k context window. SelfExtend extends Phi-2 to 4k ($G_s = 4$,$w_n = 512$), 6k ($G_s = 8$,$w_n = 512$) and 8k ($G_s = 12$,$w_n = 512$). The performance improvement compared to vanilla Phi-2 is in the parenthesis.

Context Length	2k (vanilla)	4k	6k	8k
Document QA				
NarrativeQA	4.46	6.49 (+45.52%)	8.98 (+101.35%)	12.04 (+169.96%)
Qasper	7.01	11.16 (+59.20%)	12.84 (+83.17%)	12.10 (+72.61%)
Summarization				
Gov_report	25.46	27.91 (+9.62%)	28.14 (+10.53%)	27.51 (+8.05%)
Qmsum	14.32	14.88 (+3.91%)	16.72 (+16.76%)	18.58 (+29.75%)
Few-shot Learning				
Trec	50.5	60.0 (+18.81%)	62.5 (+23.76%)	60.0 (+18.81%)
Triviaqa	74.55	84.88 (+13.86%)	82.64 (+10.85%)	81.31 (+9.07%)
Coding				
Repobench-p	54.14	56.18 (+3.77%)	56.76 (+4.84%)	57.05 (+5.37%)
Lcc	58.96	59.06 (+0.17%)	58.88 (-0.14%)	59.42 (+0.78%)

1) There is a trade-off with respect to group size in SelfExtend. Generally, both too small and too large group sizes can result in inferior performance compared to an optimal level. With a large group size, position information becomes more coarse, potentially causing performance drops. Conversely, small group sizes require SelfExtend to utilize larger position embeddings to extend the context window. These larger position embeddings are less trained compared to smaller ones. For example, in Llama-2 with its 4096 context window, the relative position 4095 accounts for only 1/2048 the frequency of the relative position 2048 in training. These under-trained relative positions can also degrade performance. This trade-off produces the 'peak' shape in the figure, indicating the extended context window differs from the ideal case described in Equation (4).

2) There is also another trade-off w.r.t. neighbor window size. With larger neighbor window sizes, there is more precise information about neighbor tokens, which is the most important. But a larger neighbor window size means SelfExtend has to use a larger group size for a long sequence, compared to using a smaller neighbor window size & smaller group size, the information about the whole sequence becomes coarse.

More results can be found in Appendix G. When the group size is so large that all distant tokens are in one group, SelfExtend degenerates into ReRoPE (e.g. group size of 32768 in Figure 10).

4.6. Performance with Varying Context Window Length

To validate SelfExtend's efficacy in enabling LLMs to utilize extended context windows, we assess Phi-2's performance across varying context lengths with SelfExtend, referencing Table 5. Across four task types from LongBench, results are generally improved with longer contexts. Notably, SelfExtend monotonically enhances performance on NarrativeQA and Qmsum. While significant improvements are observed across most datasets, a 'peak' in performance suggests a trade-off, as discussed in Section 4.5: longer contexts offer more relevant information, but the larger group sizes

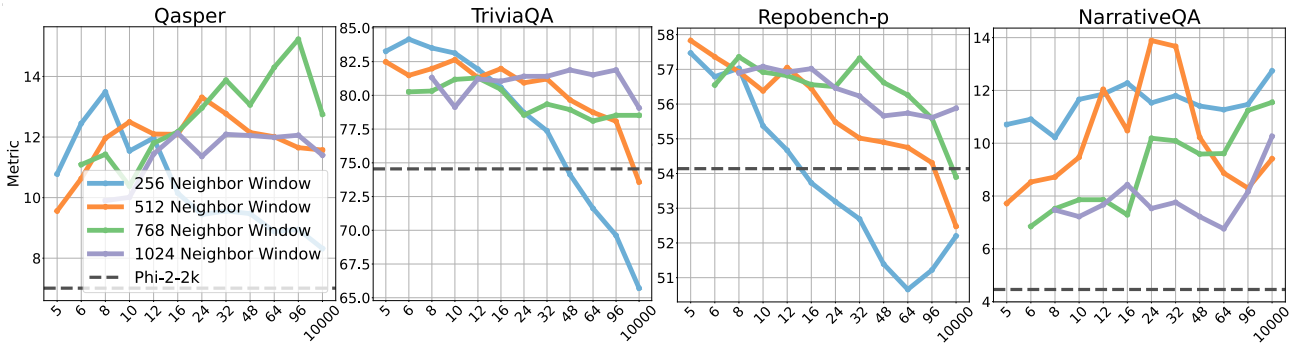

Figure 5. The performance of Phi-2 when utilizing SelfExtend to extend its context window length to 8k, with varying group sizes and neighbor window sizes. The y-axis indicates performance and the x-axis shows the group size. And neighbor window size is from $256, 512, 768, 1024$. Group size of 10000 in this experiment means all tokens out of the neighbor window are in the same group (10000>8k). Some combination (e.g. $G_s = 6$ & $w_n = 1024$) is omitted if the corresponding extended context window (Equation (4)) is smaller than 8k. The dashed line is the performance of vanilla phi-2 with a 2k context window size.

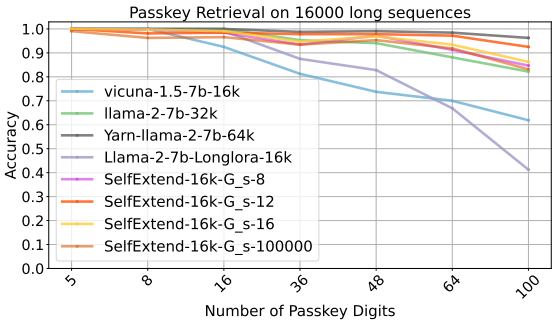

Figure 6. Passkey retrieval accuracy for four fine-tuning-based long-context models and SelfExtend on Llama-2-chat-7b across four group sizes: $8, 12, 16$, and 100000. For SelfExtend, the neighbor window is 1024. A group size of 100000 indicates that all tokens outside the neighbor window are in the same group.

required by SelfExtend to extend the context window may cause less precise positional information[4]. Regarding Lcc, performance remains consistent, possibly due to its reliance on local codes and shorter dataset lengths[5].

4.7. Varying-Length Passkey Retrieval Task

The conventional passkey retrieval task, along with prevalent benchmark datasets, primarily assesses the proficiency of LLMs in identifying and leveraging pertinent information. Traditionally, this task involves passkeys not exceeding 5 digits in length. To evaluate the LLMs' capabilities of producing consistent and precise outcomes for long sequences, we extended the task to incorporate passkeys with larger lengths. We test passkeys in $5, 8, 16, 36, 48, 64, 100$ digits. The input sequence contains $16,000$ characters. More details are presented in Appendix D.3.

The results, depicted in Figure 6, illustrate a common trend: while short passkeys of 5 or 8 digits are easily managed by all, divergences in performance emerge as the length of passkey increases. Notably, with the exception of Yarn, many tuning-based methods are unable to accurately reproduce passkeys beyond 64 digits, and some of them even experience a marked decline in performance when the passkey length exceeds 16 digits. Remarkably, although without tuning, SelfExtend maintains its superiority. These findings suggest that we should carefully choose the training approach when fine-tuning models to handle long contexts.

5. Conclusion and Discussion

In this paper, we argue that LLMs themselves have the inherent ability to handle long sequences and propose SelfExtend to elicit the inherent long context abilities for LLMs by mapping unseen relative positions into those seen during pretraining. Without any tuning or further training, SelfExtend can effectively improve LLMs' long context performance.

Limitations: SelfExtend increases computation cost with naive implementations, since it performs extra attention across all query-key pairs. However, with optimizations like blocked kernels (e.g. Flash Attention (Dao et al., 2022)), this becomes linear rather than quadratic, and the marginal cost is small enough to be ignored for long input sequences. Also, the performance degrades with large group size, preventing indefinitely long contexts. Besides, SelfExtend still processes the entire sequence to ensure information integrity, while some methods such as prompt compression (Chuang et al., 2024; Jiang et al., 2023b) can shorten the input to reduce computation. Additionally, evaluation methodologies for assessing long context abilities remain open research questions. Standard practices have yet to emerge, complicating experimental results.

Future Work: We are interested in more sophisticated mapping methods to replace the simple FLOOR operation, aiming to enhance long context understanding and extend the context window length. Additionally, we plan to further investigate the complex behaviors of LLMs using SelfExtend.

[4]Other possible reasons include: Phi-2 is a base model without instruction tuning, and SelfExtend's performance is not optimal as we use the same set of hyperparameters across all datasets, which cannot showcase SelfExtend's full potential

[5]With Phi-2 tokenizer, over 60% of Lcc instances are under 4096 tokens, with an average length of 4069.7

Impact Statement

The impact of this contribution is multifaceted. Firstly, it enhances the usability and applicability of LLMs across various domains that require the processing of lengthy input sequences, such as document analysis, long-form question answering, and retrieval augmented generation. Secondly, the ability to extend the context window without fine-tuning simplifies the deployment of advanced language models, making them more accessible to a broader range of users and applications. The availability of the code makes this solution readily accessible to researchers and practitioners, promising widespread adoption and further innovation in the field of natural language processing.

Acknowledgements

The authors thank the anonymous reviewers for their helpful comments. This work is in part supported by NSF grants NSF IIS-2310260 and IIS-2224843. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

amazon. Mistrallite model. https://huggingface.co/amazon/MistralLite, 2023. [Online; accessed 29-December-2023].

An, C., Gong, S., Zhong, M., Li, M., Zhang, J., Kong, L., and Qiu, X. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*, 2023.

Anothropic. Long context prompting for claude 2.1. https://www.anthropic.com/news/claude-2-1-prompting, 2023.

Bai, T., Luo, J., Zhao, J., Wen, B., and Wang, Q. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, G., Li, X., Meng, Z., Liang, S., and Bing, L. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*, 2023a.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023b.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023c.

Chuang, Y.-N., Xing, T., Chang, C.-Y., Liu, Z., Chen, X., and Hu, X. Learning to compress prompt in natural language formats. *arXiv preprint arXiv:2402.18700*, 2024.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

gkamradt. Llmtest_needleinahaystack: Doing simple retrieval from llm models. https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main, 2023. [Online; accessed 29-December-2023].

Han, C., Wang, Q., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Javaheripi, M., Bubeck, S., Abdin, M., Aneja, J., Bubeck, S., Mendes, C. C. T., Chen, W., Giorno, A. D., Eldan, R., Gopi, S., Gunasekar, S., Javaheripi, M., Kauffmann, P., Lee, Y. T., Li, Y., Nguyen, A., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Santacroce, M., Behl, H. S., Kalai, A. T., Wang, X., Ward, R., Witte, P., Zhang, C., and Zhang, Y. Phi-2: The surprising power of small language models, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023a.

Jiang, H., Wu, Q., , Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. LongLLMLingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *ArXiv preprint*, abs/2310.06839, 2023b. URL https://arxiv.org/abs/2310.06839.

Ke, G., He, D., and Liu, T.-Y. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

Kim, D., Park, C., Kim, S., Lee, W., Song, W., Kim, Y., Kim, H., Kim, Y., Lee, H., Kim, J., et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*, 2023.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Liu, J., Shen, Z., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

Mohtashami, A. and Jaggi, M. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

Pal, A., Karkhanis, D., Roberts, M., Dooley, S., Sundararajan, A., and Naidu, S. Giraffe: Adventures in expanding context lengths in llms. *arXiv preprint arXiv:2308.10882*, 2023.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Shi, H., Gao, J., Ren, X., Xu, H., Liang, X., Li, Z., and Kwok, J. T.-Y. Sparsebert: Rethinking the importance analysis in self-attention. In *International Conference on Machine Learning*, pp. 9547–9557. PMLR, 2021.

Su, J. Rectified rotary position embeddings. https://github.com/bojone/rerope, 2023.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. RoFormer: Enhanced transformer with rotary position embedding, 2022. arXiv: 2104.09864.

Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X., and Wei, F. A length-extrapolatable transformer. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14590–14604, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.816. URL https://aclanthology.org/2023.acl-long.816.

Team, M. N. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL www.mosaicml.com/blog/mpt-7b. Accessed: 2023-05-05.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, K., Peng, H., Chen, M., Fu, J., and Chao, H. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10033–10041, 2021.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., and Hu, X. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*, 2023.

Yin Song and Chen Wu and Eden Duthie. amazon/MistralLite, 2023. URL https://huggingface.co/amazon/MistralLite.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhang, J., Chao, H., Dhurandhar, A., Chen, P.-Y., Tajer, A., Xu, Y., and Yan, P. When neural networks fail to generalize? a model sensitivity perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11219–11227, 2023.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhu, D., Yang, N., Wang, L., Song, Y., Wu, W., Wei, F., and Li, S. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*, 2023.

A. Pseudocode of SelfExtend

Algorithm 1 PyTorch-style Pseudocode of SelfExtend

```
q, k, v # queries, keys, and values
seq_len, pos # input sequence length,
    position_idx
g_size, w_size = G, w_n

# normal self-attention
ngb_q = apply_pos_emcode(q, pos)
ngb_k = apply_pos_emcode(k, pos)
ngb_attn = matmul(ngb_q, ngb_k)
ngb_attn = causal_mask(ngb_attn)

# grouped self-attention
g_pos = pos // g_size  # the floor
    operation
shift = w_size - w_size // g_size
s_g_pos = g_pos + shift
g_q = apply_pos_emcode(q, s_g_pos)
g_k = apply_pos_emcode(k, g_pos)
g_attn = matmul(g_q, g_k)
g_attn = causal_mask(g_attn)

g_mask = tril(ones([seq_len-w_size, seq_len
    -w_size]))
mask = ones([seq_len, seq_len])
mask[w_size:, :-w_size] -= g_mask

attn = where(mask, ngb_attn, g_attn) #
    merge by replacement

attn_weights = softmax(attn)
output = matmul(attn_weights, v)
```

B. Perplexity as a Metric for Long Context Capabilities

PPL is not an effective metric for measuring the ability of LLMs to handle long contexts. In Figure 7, we introduce a seeming plausible context window extension method named 'Infinite'. When evaluated on PG19 using the same protocol, Llama-2-7b-chat with 'Infinite' achieves PPL scores that are comparable to, or even lower than, those achieved by SelfExtend, as demonstrated in Table 6. However, 'Infinite' essentially mimics the process of dividing a long sequence into short sub-sequences before processing them with LLMs, indicating that it does not genuinely address long context handling.

Table 6. Perplexity on the PG19 dataset: For 'Infinite', we set three different local window sizes: 1024, 2048, and 4096. We have also included the results from Table 1 for comparison.

Model	Evaluation Context Window Size						
Name	**4096**	**6144**	**8192**	**10240**	**12288**	**14336**	**16384**
Llama-2-7b-chat	9.181	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
SelfExtend-Llama-2-7b-chat	8.885	8.828	9.220	8.956	9.217	9.413	9.274
1024-'Infinite'–Llama-2-7b-chat	9.556	9.393	9.728	9.266	9.400	9.369	9.142
2048-'Infinite'–Llama-2-7b-chat	9.288	9.045	9.478	8.993	9.128	9.105	8.872
4096-'Infinite'–Llama-2-7b-chat	9.181	9.045	9.506	8.993	9.165	9.105	8.856
Mistral-7b-instruct-0.1 w/ SWA	9.295	9.197	9.532	9.242	9.198	9.278	9.294
Mistral-7b-instruct-0.1 w/o SWA	9.295	9.205	10.20	55.35	$> 10^3$	$> 10^3$	$> 10^3$
SelfExtend-Mistral-7b-instruct-0.1	9.272	9.103	9.369	9.070	8.956	9.022	9.128

The discrepancy between Perplexity (PPL) and long context ability primarily stems from how PPL is calculated by averaging over numerous tokens. As long as the majority of tokens are modeled accurately, PPL will remain low. This is closely related to the influence of neighboring tokens. Information from neighboring tokens—such as those within the local attention

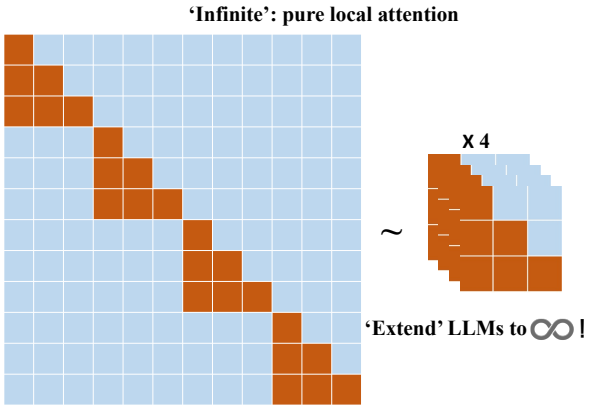

Figure 7. 'Infinite': a seemingly plausible method that extends an LLM's context window to 'infinite' length. It achieves this by dividing the entire self-attention area into multiple small, local self-attention areas. The size of the local window (i.e., the spanning range) of a local self-attention area is the sole hyperparameter for "Infinite". For instance, with a local window set to 4 and a 16-token-long input, "Infinite" essentially processes the input as four sequences of 4 tokens each.

window of 'Infinite'—can suffice for predicting most tokens, thus leading to a low PPL. However, a few critical tokens, which are crucial for understanding long contexts and answering questions, may not be predicted accurately.

Additionally, unlike the pre-training process where the cross-entropy loss corresponds directly to perplexity, measuring PPL during inference is static. It resembles a specific point on the loss curve observed during pre-training. While a decreasing trend in loss during pre-training indicates good performance, a single point on the training loss curve cannot determine the performance.

In summary, while low PPL is essential for a good model, lower PPL does not necessarily equate to better performance in understanding long contexts.

C. SelfExtend with Varying Group Size and Neighbor Window

To comprehensively understand SelfExtend's influence on LLMs, unlike previous experiments which used long context settings, we evaluate with smaller neighbor window sizes on four standard benchmark tasks: ARC-c, GSM8k, Hellaswag and MMLU. We use Phi-2 as the extended LLM. The results are shown in Figure 8. We didn't include TruthfulQA because its average length is less than 300 words, while the four datasets we used have an average length greater than 700 words. In general, SelfExtend has a minor influence on Phi-2 as long as the neighbor window size is over 128. In many cases, SelfExtend even performs slightly better than vanilla Phi-2. When the neighbor window is too small (e.g. 64 tokens), if the group size is large, as expected, the positional information loss is too high and Phi-2's performance degrades. Also, on difficult tasks such as MMLU and Helleswag, we observe a monotonic decrease in performance with increasing group size for all neighbor windows. In summary, even when applying SelfExtend to short context tasks, as long as the hyperparameters are not extreme, SelfExtend does not harm the model.

D. Detailed Experimental Setting

In this appendix, we present the details of the experiments in our paper.

D.1. Experimental Setting on Language Modeling Tasks

This is not the standard setting for PPL testing on PG-19. We use the first sentence of each book in PG19's test set (100 books) to test the language modeling ability. The results cannot be directly compared to the PPL reported by other papers. We chose this setting because our computation resources are very limited. This setting saves a lot and it can still show the behavior of LLMs w.r.t. PPL. All PPL results were calculated using the sliding window method (Press et al., 2021) with $S = 256$. We evaluated how the PPL changes as the input length increases. In Table 1, SelfExtend extends the original Llama-2's context window length from 4096 (4k) to over 16384 (16k) with group size G_s set as 8 and neighbor window w_n

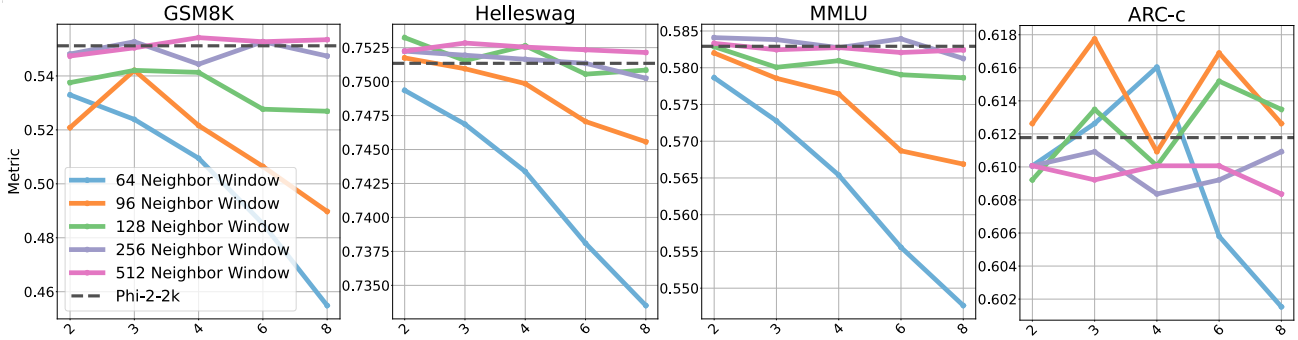

Figure 8. Phi-2 with SelfExtend on GSM8K, Helleswag, MMLU and ARC-c, compared to the vanilla Phi-2 (Phi-2-2k). The x-axis shows the group size and the y-axis indicates performance as measured by the corresponding metrics.

set as 1024 (1k). For Mistral model, without SWA, the context window is 8192 (8k) and it is also extended by SelfExtend with the same setting to larger than 16k. With SWA, Mistral can digest an infinite length of sequences and its default sliding window is 4096.

D.2. Experimental Setting on Passkey Retrieval Task

Compared to other synthetic tasks, such as "Needle in a Haystack" (gkamradt, 2023), the model's performance on this is not sensitive to the prompt (Anothropic, 2023). This may come from the fact that the sentence carrying the passkey is very different from those repeated random texts surrounding it. Empirically, within the effective context window, almost all LLMs, including those without any instruction tuning or alignment, can locate the sentence carrying the passkey. Although this task is easy and far from real-world scenarios, it tests two fundamental capabilities of LLMs: 1. The model should be able to recognize and locate the useful information across all positions of the input sequence (the most fundamental understanding capability); 2. The model should be able to use the perceived information to finish tasks (the most fundamental generation capability).

An example of passkey is as the following:

Example:

Prompt: There is an important info hidden inside a lot of irrelevant text. Find it and memorize it. I will quiz you about the important information there ..... back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again.The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again.The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again.The pass key is 60151. Remember it. 60151 is the pass key. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again.The grass is green. The sky ..... What is the passkey?

Ground Truth: 60151

Figure 9. An example of the passkey retrieval task.

D.3. Experimental Setting on Varying-Length Passkey Retrieval Task

In this experiment, we use the following models: Llama2-7b-chat with SelfExtend, LongLora-7b-16k[6],vicuna-1.5-7b-16k, Together AI's Llama-2-7b-32k[7], and Yarn-Llama-2-7b-64k.

[6]We use its fully fine-tuned variant, as we cannot use the LongAlpaca version to get reasonable performance for this specific task. For more details about the model: https://huggingface.co/Yukang/Llama-2-7b-longlora-16k

[7]Both vicuna-1.5-7b-16k and Together AI's Llama-2-7b-32k were fine-tuned using position interpolation

E. Detail of LLMs

Here, we list the links to the details of the LLMs utilized in our experiments.

Table 7. LLMs used in the experiments

Model Name	URL
Llama-2-7b-chat-hf (Touvron et al., 2023)	https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
Mistral-7B-Instruct-v0.1 (Jiang et al., 2023a)	https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1
Phi-2 (Javaheripi et al., 2023)	https://huggingface.co/microsoft/phi-2
SOLAR-10.7B-Instruct-v1.0 (Kim et al., 2023)	https://huggingface.co/upstage/SOLAR-10.7B-Instruct-v1.0
LongChat-7b-v1.5-32k	https://huggingface.co/lmsys/longchat-7b-v1.5-32k
togethercomputer/LLaMA-2-7B-32K	https://huggingface.co/togethercomputer/LLaMA-2-7B-32K
CLEX-7B-16K (Chen et al., 2023a)	https://huggingface.co/DAMO-NLP-SG/CLEX-7B-16K
CodeLlama-7b-hf (Rozière et al., 2023)	https://huggingface.co/codellama/CodeLlama-7b-hf
vicuna-7b-v1.5-16k	https://huggingface.co/lmsys/vicuna-7b-v1.5-16k
MistralLite (amazon, 2023)	https://huggingface.co/amazon/MistralLite

F. SelfExtend with Other Positional Encodings

In this section, we test SelfExtend on LLMs using non-RoPE positional encodings. We implemented SelfExtend for MPT-7b-chat (Team, 2023), which uses Alibi (Press et al., 2021) as its positional embedding method. We conducted experiments on the PG19 dataset. The results are shown in Table 8. The results show that SelfExtend is able to work with non-RoPE positional encodings, which are pretty similar to models with RoPE positional encoding,

Table 8. Perplexity of MPT-7b-chat on PG19 with different sequence length. The vanilla MPT-7b-chat has a context window of 2k tokens. For SelfExtend, We set the neighbor window as 512 and set the group size as 6.

MPT-7b-chat	1024	2048	3172	4096	5120	6144	8192
Vanilla (2k)	8.8	9.6	12.0	26.3	52.0	115.5	196.0
SelfExtend	8.9	9.9	10.6	10.8	11.1	11.3	11.6

G. Hyperparameyer Selection for SelfExtend

We conduct experiments on "Needle in a Haystack" (gkamradt, 2023), to investigate the impacts of group size and neighbor window size. The results are shown in Figure 10.

The experimental results indicate that SelfExtend is not overly sensitive to hyperparameter selection. Predefined, heuristic values for group size and neighbor window size are often sufficient to achieve satisfactory performance, as long as group size and neighbor window are not too large or too small. We conclude those results as an empirical rule. Denoting the pretraining context window as L, the target extension length as N, the neighbor window as W, and the group size as G, the empirical rule for selecting hyperparameters is to ensure that the following inequality holds:

$$\frac{1}{2} \times L > W + \frac{N - W}{G} \tag{5}$$

We believe this empirical rule is due the fact that: large relative positions are not well trained. Empirically, only a portion($\sim \frac{1}{2}$) of positions are well-trained and SelfExtend should only leverage these well-trained relative positions for the extension. This finding explains: excessively small group sizes can degrade performance, as they provide precise position information but require SelfExtend to utilize less well-trained relative positions for extension; excessively large neighbor window sizes can also degrade performance, as they provide more neighbor information but necessitate the use of less well-trained relative positions for extension.

However, the current observation may not be applicable to all models. For example, we've found that Llama3 series should use a much smaller neighbor window (~ 100). We may dive deeper to investigate the interaction among those hyperparameters and models. Besides following the empirical rule. One could use a simple and easy-to-run representative task to find proper hyperparameters.

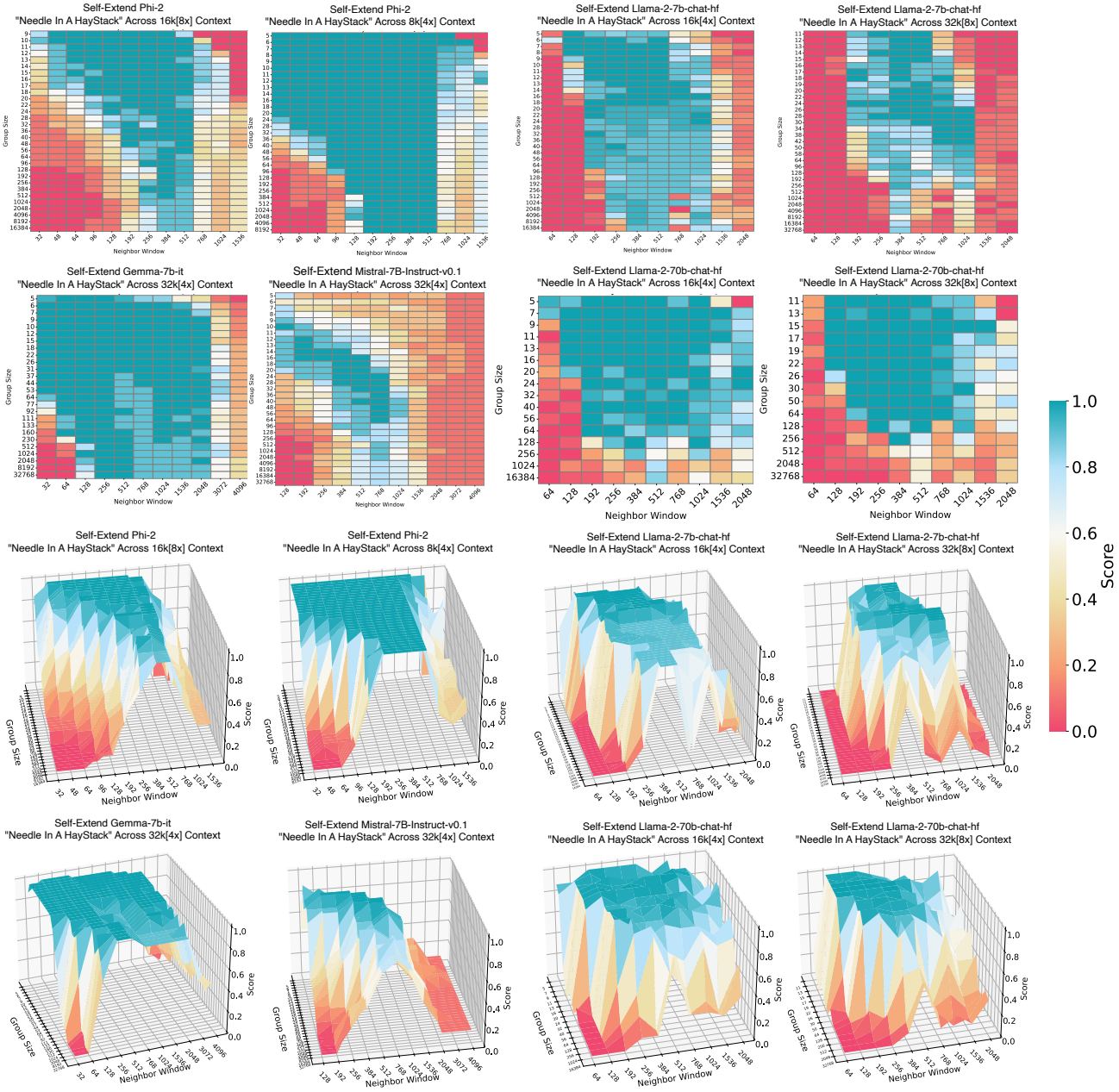

Figure 10. Impacts of group size and neighbor window size