SELF: Self-Extend the Context Length With Logistic Growth Function

Anonymous ACL submission

Abstract

Large language models suffer issues when operated on long contexts that are larger than their training context length due to the standard position encoding for tokens in the attention layer. Tokens a long distance apart will rarely have an effect on each other and long prompts yield unexpected results. To solve this problem, we propose *SELF (Self-Extend the Context Length With Logistic Growth Function)*: a solution of grouping consecutive tokens at varying group sizes using a logistic capacity equation combined with a constant group size at smaller relative distances. Our model had an increase in performance of up to 12% compared to the LongLM extension method in LEval (specifically on the Qwen model). On summarization related tasks in LongBench, our model performed up to 6.4% better than LongLM (specifically on the Llama-2-7b model). On reading comprehension tasks from LEval, our model performed up to 5.4% better than the LongLM. Our code is available at https://anonymous.4open.science/r/SELF-LLM-7705.

1 Introduction

Large language models (LLMs) are typically pretrained on sequences with fixed maximum context lengths (e.g., 2k–4k tokens), limiting their ability to reason over or generate responses based on longer inputs. When the context length exceeds the pretraining context length, the output is severely degraded and can become unreadable and undecipherable (Xiao et al., 2024b; Peng et al., 2023a; Han et al., 2024a; Chen et al., 2023b; Xiong et al., 2023). The main reason why the output is unpredictable when dealing with long context is Out-of-distribution (O.O.D) issues of the relative positional for LLMs using RoPE (Liu et al., 2023; Bai et al., 2021; Zhang et al., 2022a). When encountering relative distances on which models were not trained, model seems to generate unpredictable output vectors that cannot be decoded by the tokenizer.

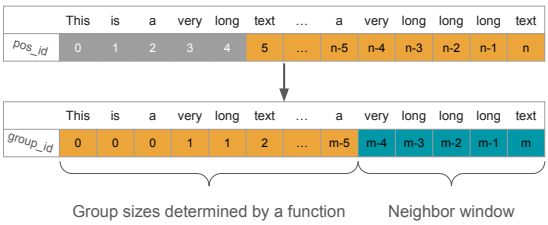

Figure 1: Illustration of our method in extending context length. Given a sequence of length n, that is larger than the training sequence length, the model groups consecutive tokens into groups whose sizes are determined by a function with the help of the neighbor window. As a result, the greatest index is now $m < n$, and the sequence now can be fully in the model's scope.

The most intuitive way is to fine-tune the models to extend the context windows, which needs high-quality long-context data and comes with a trade-off in the performance of short-context tasks (Chen et al., 2023b,b; Zhu et al., 2024). Thus, there exist some training-free methods. For example,Jin et al. (2024a) introduced Self-Extend, which leverages the model's inherent ability to generalize to out-of-distribution (O.O.D) contexts by remapping untrained relative distances to those observed during training. This is done by grouping consecutive tokens into fixed-size chunks, combined with a neighbor window for nearby tokens.

While LongLM's method shows promising results on long-context tasks, we propose a more adaptive strategy grounded in the observation that, in natural language, the relevance of a token typically decreases with its distance from the current context. This suggests that distant tokens can be grouped into larger units without significantly harming comprehension. Based on this intuition, we introduce a dynamic grouping strategy where group sizes increase with distance from the query. Unlike fixed-size chunking, our approach determines group boundaries through a distance-aware

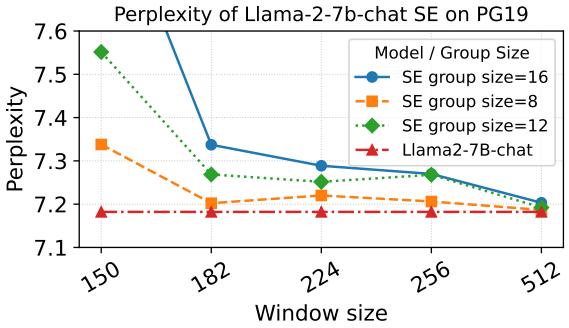

Figure 2: Illustration about relation between neighbor window and perplexity after applying Self-Extend (Jin et al., 2024a). The results is derived from testing Llama-2-7B and its Self-Extend variants on the first book in PG19 (Rae et al., 2019) with sequences of 2048 tokens. The perplexity of models applying Self-Extend slowly approaches the perplexity of the original model when increasing the neighbor window size.

function, allowing for more efficient use of context length while preserving semantic fidelity.

Thus, we introduce *SELF (Self-Extend the Context Length With Logistic Growth Function)*, a more adaptive token grouping strategy that dynamically adjusts group boundaries based on context structure rather than relying on fixed-size chunks. This allows for better capture of long-range dependencies and finer preservation of semantic boundaries with different distances. In essence, our method addresses the O.O.D. challenge through a shared principle but differs in the way token groups are constructed (see Figure 1 [1] for illustration).

Through comprehensive experimental results, we witness an increase up to 8% (specifically Qwen2-7B model (Yang et al., 2024)) of accuracy when applying our grouping method compared to Self-Extend (Jin et al., 2024a) when benchmarking on LEval. We also witnessed an accuracy increase of up to 5% when using SELF over LongLM on the Llama-2-7B (Touvron et al., 2023b).

2 Motivations

LongLM (Jin et al., 2024a) proposes a solution to handle prompts that are longer than the models' pretraining sequence lengths by grouping tokens at far distances because the exact position is less important than the relative order of information in long context and keep the exact positions of closer token by a neighbor window. However, the output's perplexity will increase right after where the neighbor window ended. As a result, to make the model less "confused", the neighbor window has to be increased (see Figure 2), which decreases the total number of tokens the models after applying Self-Extend can handle.

Although LongLM's method yielded significantly improved results in key retrieval, the LongBench benchmark (Bai et al., 2024a) and the LEval benchmark (An et al., 2023), we believe that we can increase the total context length by leveraging a property of natural language: the farther a word is from the current token, the less important it tends to be to the current context.

In LongLM (Jin et al., 2024a), the group size is the same for every group, which is not the most optimized way. Intuitively, in natural languages, the further a token, the less relevant the token is to the context, allowing us to group them into progressively larger groups, meaning that we can improve the total context length by allowing larger groups without significantly trade-off in model's comprehension ability. By this intuition, the group sizes have to be dynamic, which means that they have to be determined by a function.

Therefore, we need to choose a monotonic increasing function for group sizes (the further, the larger the group). However, if the group size is too large, it will affect the performance because every word is treated the same regardless of their positions. Therefore, we must choose a function whose maximum value is limited and controllable.

Based on those conditions, we decided to choose the Logistic growth Function, which is a monotonic increasing function with a defined maximum value. Because the group sizes have to be integers, we will take the floor of the Logistic growth Function.

$$f(x) = \lfloor \frac{Ce^{rx}}{C + e^{rx} - 1} \rfloor$$

, C is the capacity i.e. the maximum group size and r is the growth rate of group sizes.

3 Preliminaries

3.1 Position Encoding

Most models use two types of position encodings, relative and absolute position encoding. Relative position encoding utilizes the distance between one token and another token while absolute position encoding uses the token's position from 0. (Vaswani et al., 2023) Since the importance of words is usually based on how far they are away from the base word, relative position encoding is

[1] This is an oversimplification of how the method works. More details will be explained in Our Proposal section

more common. Examples of absolute encoding include GPT3, learned positional encoding (Brown et al., 2020), OPT (Zhang et al., 2022b). Examples of relative encoding include T5, learnable attention bias (Xue et al., 2021), Transformer-XL (Dai et al., 2019), Alibi, fixed linear attention (Press et al., 2022). This is especially important when it comes to long context prompts as our LLM might need to consider tokens further away as still being important. These position encodings are applied at the attention layer so that when tokens are interconnected with eachother the positions are considered. Our goal is to design a mechanism where we can consider tokens far apart in our decision making while also holding closer tokens to a high importance. Considering an example of long context key retrieval, we need to consider the close-by tokens (instructions) to a high degree but also ensure the key (at a far away position) is considered.

3.2 RoPE

Considering tokens $a_1 \dots a_n$ and their embeddings $x_1 \dots x_n$ where each embedding is a real matrix. RoPE (Su et al., 2023) integrates the position information into the query and key vectors. If done properly, $q^T k$ will already contain the positional embeddings preventing an extra step from being needed. To embed the position, RoPE uses the function $q_m = f_q(x_m, m) \in \mathbb{R}^{|n|}, k_n = f_x(x_n, n) \in \mathbb{R}^{|n|}$ where $|L|$ is the hidden dimension of each head. $f_q(x_m, m) = W_q x_m e^{im\theta}, f_k(x_n, n) = W_k x_n e^{in\theta}, \theta_d = b^{-2d/|D|}$. The positional embedding system keeps the real section of $q^T k$ which is $\text{Re}(q^* k)$. The dot product of the query and key will always yield a result depending on the relative distance between the two tokens as follows:

$$\begin{aligned} &\langle f_q(x_m, m), f_k(x_n, n)\rangle_{\mathbb{R}} \\ &= \text{Re}(\langle f_q(x_m, m), f_k(x_n, n)\rangle_{\mathbb{C}}) \\ &= \text{Re}(x_m^* W_q^* W_k^* x_n e^{i\theta(m-n)}) \\ &= g(x_m, x_n, m-n), \end{aligned} \tag{1}$$

where g is an abstract mapping function.

4 Our Proposal

4.1 Self-Extend with constant group size

Self-Extend(Jin et al., 2024a) maps unseen relative positions to trained relative positions by using the FLOOR operation to group neighboring tokens into one single group that shares the same positional index.

Their important finding is the importance of neighbor attention. By just purely grouping tokens together, the perplexity will be higher than in the original model. Grouping all tokens with a constant group size degrades the effect of closer tokens which usually have more importance. To solve this problem, LongLM uses separate grouped attention, for tokens further away, and neighbor attention, for nearby tokens. Acknowledging this, our method will also apply neighbor attention.

4.2 SELF: Self-Extend with dynamic group size

Despite successfully tricking the LLM into believing that the tokens are closer than they really are, LongLM's approach abruptly increases the group size from 1 (within the neighbor window, the group size can be regarded as 1) to a much larger value (the value of group size, e.g., 512). To avoid this sudden jump, we propose that group sizes should increase gradually rather than all at once. More specifically, the group size should grow according to a smooth function such as the Logistic Growth Function, which starts small and increases steadily. Based on this idea, we propose a new method called *SELF (Self-Extend the Context Length With Logistic Growth Function)*.

In SELF, we use a function $f : \mathbb{N} \to \mathbb{N}$ to determine the size of each group. Given a group index (like the 0th, 1st group), this function returns the number of tokens assigned to that group.

Example 1: Given a function f whose $f(0) = 1, f(1) = 2, f(2) = 2, f(3) = 3$ and $f(4) = 3$. The grouping will be:

$$F = [0, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4]$$

Let's define:

- $G^K : \mathbb{N} \to \mathbb{N}$ as the group position index used in the encoding of the key-value pairs.

$$G_i^K = F_i \tag{2}$$

- $G^Q : \mathbb{N} \to \mathbb{N}$ as the group position index used in the encoding of the query.
- $R : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is the relative distance between G^Q and G^K.

The relative position right after the neighbor window ($R_{i,i-W}$) has to be W, where W is the width of the neighbor window, because the relative positions inside the neighbor window will range from 0

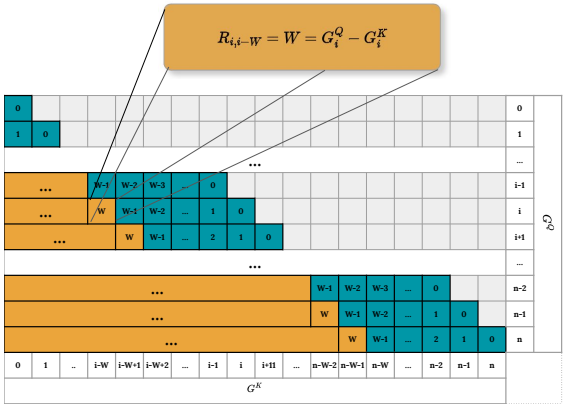

Figure 3: Illustration of the relation between G^K and G^Q knowing that the relative position right after the neighbor window has to be W.

to $W-1$ (see illustration in Figure 3). Therefore,

$$G_i^Q = \begin{cases} W + G_{i-W}^K, & \text{if } i \leq W \\ c, & \text{otherwise} \end{cases} \quad (3)$$

No matter what constant c we choose, it will be completely covered by the neighbor window.

If we used only group attention, the number of tokens can be fully extended to $\sum_{i=1}^{L} f(i)$. However, because we have the neighbor window, $R_{n,n-W} = W$ instead of $R_{n,n-W} = \max(F) - F_W$, that is, it takes $W + F_W - \max(F)$ more indices than using only group attention. Therefore, the number of tokens can be extended to

$$L' = \sum_{i=1}^{L+\max(F)-W-F_W} f(i)$$

where L is the initial token limit.

This formula gives the total number of tokens that can be processed using our SELF method, which blends regular and group-based attention in a way that grows group sizes smoothly and avoids any sudden jumps that could disrupt the model.

4.3 Efficient Implementation: grouping indices in parallel

The most native approach to calculate F given f is to start with an empty F, than sequentially compute $f(i)$ and add $f(i)$ more elements to the end of F.

$$F \leftarrow F \parallel \text{replicate}(i, f(i))$$

However, since we have thousands of tokens, computing the new positional embeddings sequentially would take $O(n)$ of run time.

Algorithm 1 Construct group indices(n, W, C, r)

$p \leftarrow -1$

for $k \leftarrow 1$ to $C-1$ **do**
  Compute parallelly $F[p+1..p+g(k)]$ using Equation (6)
  $p \leftarrow p + g(k)$
end for

Compute G^K parallelly using Equation (2)
Compute G^Q parallelly using Equation (3)

return $group\_key\_id$ and $group\_query\_id$

In order to solve this high computing time issue, we use inverse function of the grouping function which will divide each sequence into sections in which the group sizes are the same so we can easily calculate and assign group sizes in parallel. We define the inverse function $f^{-1} : \mathbb{N} \to \mathbb{N}$, when given the group size, the inverse function returns the smallest index that has the given group size.

Using Example 1, the inverse function will be $f^{-1}(1) = 0, f^{-1}(2) = 1$ and $f^{-1}(3) = 3$. Let's define the function $g : \mathbb{N} \to \mathbb{N}$, when given the group size, the function returns the total number of elements that are in the group of the given size.

$$g(x) = x \cdot [f^{-1}(x+1) - f^{-1}(x) + 1] \quad (4)$$

In order to find the F_i given i, define k the largest number such that:

$$S = \sum_{j=1}^{k} g(j) < i \quad (5)$$

This means that F_i has to be in the group whose size is $k+1$ and it is $\lceil \frac{i-S}{k+1} \rceil$ indices away from the last index of the group whose size is k, which is $f^{-1}(k+1) - 1$. Therefore,

$$F_i = [f^{-1}(k+1) - 1] + \lceil \frac{i-S}{k+1} \rceil \quad (6)$$

Now, considering the logistic growth function, we have:

$$f(x) = \lfloor \frac{Ce^{rx}}{C + e^{rx} - 1} \rfloor$$

$$f^{-1}(y) = \lfloor \frac{\ln(Cy - y) - \ln(C - y)}{r} \rfloor$$

In the logistic growth function, the maximum group size k in Equation (5) is C, which is a very small number compared to the sequence length. We can utilize GPU parallelism using the pseudo-code.

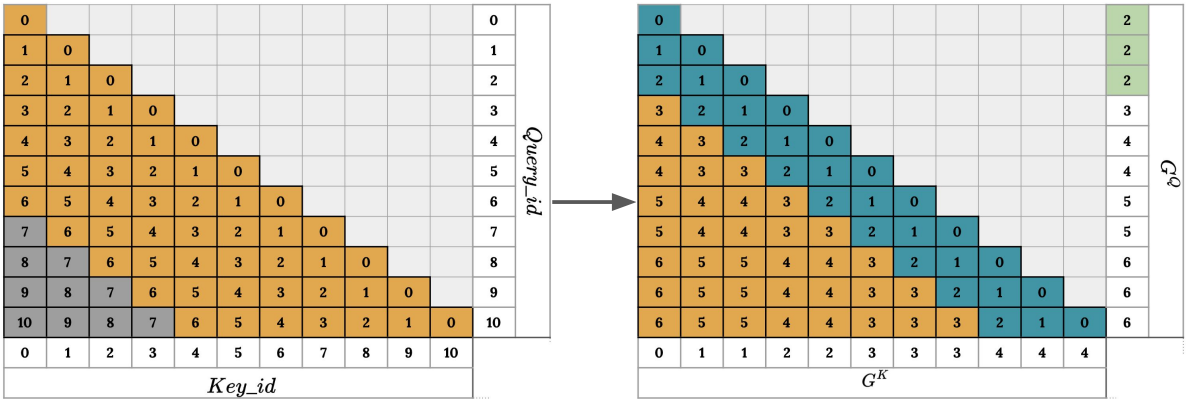

Figure 4: Illustration of the algorithm grouping the indices using the function $f : \mathbb{N} \to \mathbb{N}$, where $f(0) = 1, f(1) = 2, f(2) = 2, f(3) = 3$ and $f(4) = 3$. The sequence with length of $n = 11$ was run the model with the pretraining sequence length of $L = 6$. The numbers denote the relative position between the corresponding key and query token. It has two kinds of self-attention, similar to Self-Extend (Jin et al., 2024a): neighbor tokens inside the neighbor window ($W = 3$) (blue cells in the figure) use regular self-attention; group tokens outside the neighbor window (orange cells in the figure) use group self-attention (group indices are denoted as the G row and column in the figure). Green G^Q means it can be anything as it is covered completed by the neighbor window.

By applying Equation (4), $g(k)$ can be computed in $O(1)$. We can easily tell that the total work for computing F is $O(n + C)$, and the total work for computing G^K and G^Q knowing F is $O(n)$, since it takes $O(1)$ at each index.

Putting this together, the total work for the algorithm is $O(n + C)$ and the parallel span is $O(C)$. Assuming that there are P threads available, the runtime is bounded by:

$$T(P) = O(\max\{\frac{n + C}{P}, C\})$$

This means that if having sufficient resources (P is large enough), we can speed up to near-linear since the lower bound is C, which is usually a very small number.

5 Experiments

In this section, we first analyze the impact of *Group Size* on the SELF method based on perplexity results from the PG19 dataset, in order to identify an appropriate group size configuration. We then compare SELF with the standard SE method on real-world long-context benchmarks such as LONGBENCH and LEVAL, demonstrating the effectiveness of SELF on practical long-conetxt tasks.

We ran experiments on Llama-2-7B, Llama-2-13B (Touvron et al., 2023a), Qwen-7b (Yang et al., 2024), and a distilled reasoning model from Deepseek-R1 (DeepSeek-AI et al., 2025). We conducted our tests on Longbench v1 (Bai et al., 2024a) and LEval (An et al., 2023)[2].

5.1 Understanding the impact of group size on SELF via Perplexity

We begin by measuring the perplexity of LLaMA-2-7B-Chat with both SE and SELF under varying group sizes and context window lengths on the PG19 dataset. From Table 1, we can observe:

- SELF achieves lower perplexity scores when working with the same group size.
- When group size is small, there isn't much difference in perplexity score between SE and SELF.
- The larger the group size, the longer the sequence lengths over which noticeable differences can be observed. For example, the difference is still noticeable when $C = 32$, and the sequence length is 6144 and when $C = 64$ and the sequence length is 12288. With larger group size, it takes longer sequence to reach its maximum group size, meaning that most of the groups has their sizes less than the maximum group size, decreasing the final perplexity score.
- When dealing with sequences that are significantly longer than the original context length, the scores are basically the same for SE and SELF. When the sequence is significantly long, the amount of intermediate group is negligible compared to the number of groups that have reached the maximum group size. As a result, the model behavior closely resembles that of SE, where all groups have the maximum group size.

From the above observations, there is a trade off

[2]We skipped the GSM benchmark as we were unable to replicate the results the paper provided on our own

Table 1: Perplexity on dataset PG19 (Rae et al., 2019) first book with Llama-2-7b-chat and compare SE and SELF (the growth rate $r = 0.02$) with the same group size ($C = 16$, $C = 32$ and $C = 64$) and neighbor window.

	Model Name	Perplexity with Context Window Size (log scale) 4096	6144	8192	10240	12288	14336	16384
	Llama-2-7b-chat	7.231	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
$C = 16$	SE-Llama-2-7b-chat	7.103	7.086	7.126	7.174	7.229	7.248	7.270
	SELF-Llama-2-7b-chat	7.085	7.085	7.122	7.168	7.203	7.234	7.270
$C = 32$	SE-Llama-2-7b-chat	7.141	7.184	7.199	7.314	7.346	7.410	7.496
	SELF-Llama-2-7b-chat	7.119	7.133	7.196	7.275	7.345	7.408	7.484
$C = 64$	SE-Llama-2-7b-chat	7.186	7.316	7.303	7.458	7.530	7.625	8.041
	SELF-Llama-2-7b-chat	7.135	7.180	7.267	7.364	7.467	7.619	8.068

in perplexity when increasing the group size (Jin et al., 2024a). When using larger group size, models are more uncertain in their predictions. However, when dealing with the same group size[3], ideally, models with SELF have lower perplexity than ones with SE because instead of increasing rapidly from one to the maximum group size, models with SELF have to go through smaller group sizes (intermediate groups) before reaching the maximum group size. Therefore, although the context length after extension of both methods are approximately the same with the same group size, SELF performs better when the sequence length is not excessively larger than the orginial context window.

Base on this analysis, we decided to choose larger group size for most of our experiments compared to ones in LongLM (Jin et al., 2024a) without experiencing a significant trade off in performance.

5.2 Comparisons of SELF and SE in Real-World Long Context Tasks

5.2.1 LongBench

We conducted experiment on LongBench (Bai et al., 2024a) using Llama-2-7B and then compared our results with the original model and the model where Self-Extend is applied. Here we decided instead of using small group size of 6 and 8 like in LongLM (Jin et al., 2024a), we used a much bigger group size ($C = 32$) and still observed a better results in most tasks. The results are in Table 2 We see an improvement of summarizing tasks because having to go through smaller group sizes makes the models have a better understanding of the text. Moreover, our suspicion that SELF tends to perform better when the length is closer to the original context

[3]Group size in context of SELF refer to the maximum group size i.e. the capacity in the Logistic growth function

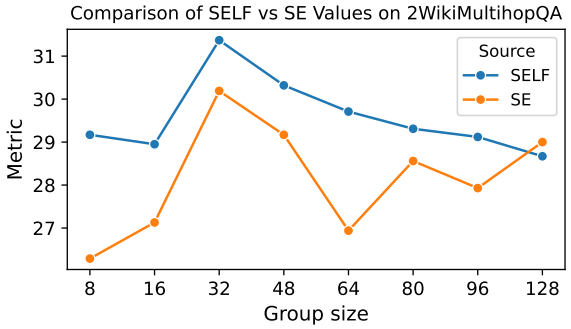

Figure 5: Compare the trade off with different group sizes of SE and SELF on 2WikiMultihopQA. The two grouping methods has the same neighbor window size $W = 1024$.

window is also confirmed since SELF performs better in tasks where the average context length is not significantly long such as MultiFieldQA, 2WikiMultihopQA, HotpotQA and TREC.

We can also observe the trend in trade off of accuracy in 2WikiMultihopQA task (see Figure 5). Unlike SE, whose changes are more abrupt and unpredictable despite of the general trend is still decreasing (this can also be observed when SE doing other tasks with vary group sizes (Jin et al., 2024a)), SELF's accuracy seems to slowly decrease after reaching its peak. In SE, group indices can be every different with different group sizes. In contrast, SELF's group sizes increase gradually, resulting in overlapping early groups. As the result, the differences between varying group sizes are relatively subtle compare to SE.

5.2.2 LEval

We tested one 4 different models using the same hyperparameter as in the LongBench v1 test (see the result Table 3).

- **Llama-2-7B:** An improvement can be seen in every tasks except for CodeU. However, the dif-

Table 2: Performance of different models on LongBench (Bai et al., 2024a). * means that the results are reported by Self-Extend (Jin et al., 2024a), * means that the results are run by us (single run). The suffix number (e.g. '25k') indicates the maximum context window of the model. The 'SE' prefix indicates Self-Extend is applied to this model and the prefix 'SEF' indicates that our Self-Extend with Logistic Growth Function is applied. The best performance in each section will be 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
SE vs SELF	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
	SELF-Llama-2-7B-chat-100k*	17.4	25.74	**37.61**	**36.30**	**31.37**	13.11	**27.9**	**21.81**	**27.54**	**69.50**	76.97	40.85	**6.16**	6.0	60.49	51.55
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
	SELF-Llama-2-7B-chat-100k*	17.4	25.74	37.61	**36.30**	**31.37**	13.11	27.9	21.81	**27.54**	69.50	76.97	40.85	**6.16**	6.0	60.49	51.55
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

ference in CodeU is not significant as the most correct models could answer only 1 correctly compare to 0 for model that apply SELF.

- **Llama-2-13B:** the SELF version seems to perform worse than the SE version in all tasks. We could not come up with the reason why there is such a difference despite Llama-2-7B and Llama-2-13B having the same architecture.

- **Qwen-7B:** There is a significant improvement compared to SE. There is an improvement compared to the raw model, but not significant.

- **Deepseek-R1-Distill-Qwen-7B**[4]: For this reasoning model, we forced the model to reason before giving answer by adding the open tag `<think>`. An significant decrease in the accuracy of models after applying Self-Extend (both SE and SELF) can be observed. We suspect the Reinforcement Learning process to improve reasoning ability does have influence on this effect as reinforcement learning was applied on exact position and the model's reasoning ability wasn't optimized for group attention. We also witnessed models applying SE and SELF often stuck in reasoning loop which did not happen for the original model. However, this needs to be researched on more before making conclusion.

[4]We modified the evaluation function for MCQ. After the reasoning process, models often start their conclusion with "Answer:", in which the original code (An et al., 2023) will assume the answer to be "A" because "A" is the first capitalized letter

6 Related Works

Long context models. Many recent large language models support extended context lengths, such as GPT-4 (Achiam et al., 2023), Claude, Qwen (Yang et al., 2025a), LLaMA (Grattafiori et al., 2024), and Phi (Abdin et al., 2024). Notably, models like `Qwen2.5-7B-Instruct-1M` (Team, 2025; Yang et al., 2025b) and Llama-3.1-Nemotron-8B-UltraLong-1M-Instruct (Xu et al., 2025) are capable of handling context windows up to 1 million tokens, enabling long-range reasoning across extremely lengthy inputs. Multiple long context models exist optimized for long queries. LongChat (Li et al., 2023a; Bai et al., 2024b) is a long context LLM designed for long context conversations. To train their model, LongChat is tested against their own long context testing suite and is trained with a context size of 32K. CLEX (Chen et al., 2023a) is a long context LLM that works by using differential equations to scale positional embeddings to better support longer prompts. CodeLlama (Rozière et al., 2024) is a LLM model based on Llama 2 optimized for long context prompt performance. CodeLlama works by training the model on a longer context length of 16K.

Long context extension methods. Most models increase the context length through fine-tuning, which still does not solve the problem of attention having a minimal affect at large relative distances. To solve this problem, other extension methods use a similar system where the position encodings modify the relative positions. Models with

Table 3: Performance comparison of Llama2-7B, Llama2-13B, Qwen-7B and DeepSeek-R1-Distill-Qwen-7B before and after applying SE and SELF on LEval (An et al., 2023). The figure also includes performance of other fixed models on LEval. * means that the results are reported by LongLM (Jin et al., 2024a), * means that the results are run by us (single run). The best performance between the original, SE and SELF will be in **bold**.

Model	Coursera	TOEFL	QuALITY	CodeU	SFiction	Avg.
Llama-2-7b-chat*	29.21	51.67	37.62	**1.11**	60.15	35.95
SE-Llama-2-7b-chat*	35.76	55.39	**41.09**	**1.11**	57.81	38.23
SELF-Llama-2-7b-chat*	**36.19**	**56.88**	**41.09**	0.00	**60.94**	**39.02**
Llama-2-13b-chat*	35.75	60.96	**42.57**	**1.11**	**60.15**	40.11
SE-Llama-2-13b-chat*	**38.95**	**66.17**	41.09	**1.11**	60.15	**41.49**
SELF-Llama-2-13b-chat*	37.93	64.31	39.11	0.00	57.03	39.68
Qwen-7B*	52.18	79.18	65.35	0.00	**63.28**	52.00
SE-Qwen-7B*	53.20	78.07	59.41	0.00	57.03	49.54
SELF-Qwen-7B*	**53.34**	**80.67**	**66.83**	**4.44**	62.5	**53.56**
Reasoning Model						
DeepSeek-R1-Distill-Qwen-7B*	**58.43**	**66.54**	**48.01**	2.22	60.16	**47.07**
SE-DeepSeek-R1-Distill-Qwen-7B*	54.21	66.17	40.59	**6.66**	**62.4**	45.81
SELF-DeepSeek-R1-Distill-Qwen-7B*	40.27	58.74	37.5	1.11	50.78	37.68
Fixed Models						
Claude1.3-100k*	60.03	83.64	73.76	17.77	72.65	65.97
GPT-4-32k	75.58	84.38	82.17	25.55	74.99	73.11
Turbo-16k-0613*	63.51	78.43	61.38	12.22	64.84	60.73
Chatglm2-6b-8k*	43.75	53.90	40.59	2.22	54.68	34.69
XGen-7b-8k (2k-4k-8k)*	26.59	44.23	35.15	1.11	48.43	26.41
Chatglm2-6b-8k*	42.15	54.64	44.05	2.22	54.68	35.95
Chatglm2-6b-32k*	47.81	55.01	45.04	2.22	57.02	39.01
XGen-7b-8k*	29.06	42.37	33.66	3.33	41.40	27.63
MPT-7b-65k*	25.23	17.84	25.24	0.00	39.06	19.22

different context extension methods and their performance is mentioned in the above section above specific model performance. These include RoPE-based techniques such as Position Interpolation (PI) (Chen et al., 2023c), NTK (Peng and Quesnelle, 2023), YaRN (Peng et al., 2023b), and Self-Extend (Jin et al., 2024b); attention-architecture-based methods such as StreamingLLM (Xiao et al., 2023), LM-Infinite (Han et al., 2024b), LongLoRA (Chen et al., 2023d), Inf-LLM (Xiao et al., 2024a), and Landmark (Mohtashami and Jaggi, 2023); as well as retrieval- and compression-based approaches such as Retrievers (Xu et al., 2023), LongLLMLingua (Jiang et al., 2023), and context compression (Li et al., 2023b).

7 Conclusion

We are successfully able to implement group attention with a custom function and apply it with a logistic growth function. From our analysis, we can conclude that SELF works better than SE when dealing with the same capacity, and SELF's behavior when increasing group size is more predictable.

Our logistic capacity model for grouping tokens yielded minor to major increases in most tests across LEval[5]. The method performed best on Llama-7b and Qwen-7B. On LongBench, our method performed better across most tests or saw only minor decreases in performance. By grouping tokens using a combined constant and logistic growth positional embedding layer, we allow the LLM to consider tokens at a far distance while keeping nearby tokens more relevant. SELF increases LLM prompt performance without sacrificing runtime performance nor modifying the prompt.

[5]With the exception of CodeU, a test where all methods performed poorly

Limitations

- Although theoretically, SE and SELF has the basically same runtime complexity, SELF requires more complicated computations like $\ln$ instead of just FLOOR. As a result, running SELF takes a longer time than running SE.
- The grouping method was not tested in the variety of LLMs (only tested on LLama2-7B, Llama2-13B, Qwen-7B and Deepseek-R1-Distill-Qwen-7B) nor the variety of benchmarks.
- On reasoning models, running SELF the model would sometimes get stuck in a loop while thinking causing an unpredictable answer. This led to degraded performance compared to the raw model and SE.
- SELF still struggled on the CodeU benchmark compared to other models and would sometimes produce nonsensical outputs.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *Preprint*, arXiv:2307.11088.

Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. 2021. Recent advances in adversarial training for adversarial robustness. *Preprint*, arXiv:2102.01356.

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. 2024a. Longbench: A bilingual, multitask benchmark for long context understanding. *Preprint*, arXiv:2308.14508.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2024b. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2023a. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*.

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

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023c. 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. 2023d. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *Preprint*, arXiv:1901.02860.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024a. Lm-infinite: Zero-shot extreme length generalization for large language models. *Preprint*, arXiv:2308.16137.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024b. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008.

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

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

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

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

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023b. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2023. Towards out-of-distribution generalization: A survey. *Preprint*, arXiv:2108.13624.

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

Bowen Peng and Jeffrey Quesnelle. 2023. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

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

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

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. *Preprint*, arXiv:2108.12409.

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 Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Qwen Team. 2025. Qwen2.5-1m: Deploy your own qwen with context length up to 1m tokens.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023a. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024b. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, and 2 others. 2023. Effective long-context scaling of foundation models. *Preprint*, arXiv:2309.16039.

Chejian Xu, Wei Ping, Peng Xu, Zihan Liu, Boxin Wang, Mohammad Shoeybi, and Bryan Catanzaro. 2025. From 128k to 4m: Efficient training of ultra-long context large language models. *arXiv preprint*.

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. *arXiv preprint arXiv:2310.03025*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. *Preprint*, arXiv:2010.11934.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, and Jialong Tang... 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025b. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*.

Jiajin Zhang, Hanqing Chao, Amit Dhurandhar, Pin-Yu Chen, Ali Tajer, Yangyang Xu, and Pingkun Yan. 2022a. When neural networks fail to generalize? a model sensitivity perspective. *Preprint*, arXiv:2212.00850.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022b. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2024. Pose: Efficient context window extension of llms via positional skip-wise training. *Preprint*, arXiv:2309.10400.