SkyLadder: Better and Faster Pretraining via Context Window Scheduling

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent advancements in LLM pretraining have featured ever-expanding context windows to process longer sequences. However, our controlled study reveals that models pretrained with shorter context windows consistently outperform their long-context counterparts under a fixed token budget. This finding motivates us to explore an optimal **context window scheduling** strategy to better balance long-context capability with pretraining efficiency. To this end, we propose SkyLadder, a simple yet effective approach that implements a short-to-long context window transition. SkyLadder preserves strong standard benchmark performance, while matching or exceeding baseline results on long-context tasks. Through extensive experiments, we pretrain 1B-parameter models (up to 32K context) and 3B-parameter models (8K context) on 100B tokens, demonstrating that SkyLadder yields consistent gains of up to 3.7% on common benchmarks, while achieving up to 22% faster training speeds compared to baselines.

1 Introduction

The evolution of language models has been marked by a consistent expansion in context window sizes (Figure 1 left). While early models like GPT [39] and BERT [8] were limited to context windows of 512 tokens, subsequent models have pushed significantly beyond these bounds. GPT-2 [40] doubled this capacity to 1024 tokens, and with Large Language Models (LLMs) exceeding 1B parameters, this trend has continued: Llama [50] has a 2048-token window, followed by Llama-2 [51] (4096 tokens), and Llama-3 [13] (8192 tokens). The need for models to handle longer sequences during inference has fueled the rush to expand the context window. As models pretrained with longer context windows reduce document truncation and preserve coherence [9], there is a widespread belief that such models should perform comparably to, or even surpass, their shorter-context counterparts.

We question the common belief that larger context windows do actually improve performance. Close inspection of previous work reveals that there has yet to be a fair experimental setup for comparing models across different context windows while adhering to a fixed token budget. Using tightly controlled experiments, we test how changing only the context window size during pretraining impacts their performance. As shown in Figure 1 (right), our results indicate that models pretrained using shorter contexts always outperform long-context models, when assessed by their average performance across popular benchmarks. In addition, we verify that the performance gap is not eliminated by using advanced document packing strategies [13, 9, 44].

To ensure the model can ultimately process long sequences, the model still needs to be exposed to long sequences. However, given the finding that shorter context windows enhance performance on downstream tasks, we face a trade-off between long-context capability and pretraining effectiveness. We propose SkyLadder, a simple yet effective **context window scheduling** strategy designed to balance both objectives. SkyLadder does this by progressively expanding the size of the context

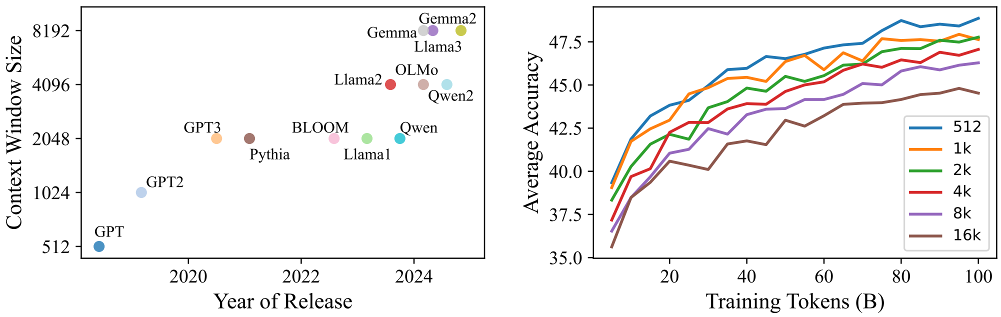


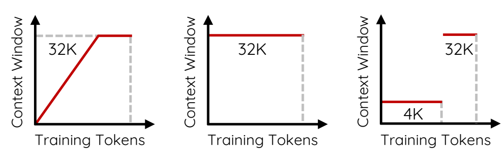
Figure 1: Left: Pretraining context window of LLMs grows over the recent years. Right: Average performance (in %) across nine downstream tasks for 1B-parameter models with different pretrained context window sizes (color-coded). Increasing the context window degrades the overall performance.

window during pretraining, beginning pretraining with a minimal short context window (e.g., 8 tokens) and progressively expanding it to the long target context window (e.g., 32,768 tokens).

Empirical results on 1B-parameter models (up to 32K context window) and 3B-parameter models (up to 8K context window) on 100B tokens demonstrate that SkyLadder outperforms naive long-context pretraining baselines, in both short- and long-context evaluation tasks. For example, models trained with SkyLadder demonstrate significantly higher accuracy on standard benchmarks (e.g., HellaSwag), and reading comprehension tasks (e.g., HotpotQA), while still maintaining competitive performance on long-context evaluations like RULER. We further investigate the mechanisms behind the superior performance by observing the training dynamics, and discover that SkyLadder exhibits more concentrated and effective attention patterns.

Overall, we suggest that the length of the context window is an important dimension in pretraining and should be scheduled over the course of training. We recommend a progressive approach that begins with a small context of 8 tokens and gradually increases according to a linear function of training steps. Given a target context window (e.g., 32K), we suggest that allocating approximately 60% of the total training tokens to this expansion phase leads to stronger downstream performance compared to baselines. This scheduling strategy optimally enhances both training efficiency and model capability, offering a practical recipe for improving pretraining in language models.

2 Related Work



(a) context window scheduling **(ours)** (b) constant long context (c) continual pretraining on long context

Figure 2: Schematic comparison of training-time context window scheduling.

Context Window Scheduling. Early work explored gradually increasing the context window in smaller models like BERT and GPT-2, to improve training stability and efficiency [35, 28, 21]. Notably, Li et al. [28] proposed length warmup for more stable training but did not show clear performance gains, while Jin et al. [21] focused on training acceleration in 400M models. We extend these findings by demonstrating, for the first time, that context window scheduling significantly boosts both *efficiency* and *performance* at much larger scales (up to 3B parameters). A parallel approach from Pouransari et al. [38] segments training documents by length, but Fu et al. [10] caution that such segmentation can introduce domain biases, as longer texts often cluster in specific domains such as books. Recent developments in continual pretraining with long context windows [37, 54, 12], can also be viewed through the lens of context window scheduling with different strategies (illustrated in Figure 2). Our work represents the first demonstration of both effectiveness and efficiency of context window scheduling, providing empirical evidence of its benefits in both standard and long-context benchmarks.

Long-Context Language Models. Long-context language models have received a lot of attention due to their ability to capture extended dependencies across large textual windows. Most existing approaches follow a *continual pretraining* paradigm [10, 56], which extends a pretrained backbone model to longer contexts through specialized fine-tuning or additional training. Several works propose to intervene in the positional embeddings to accommodate longer sequences [1, 31, 37, 4, 22], while

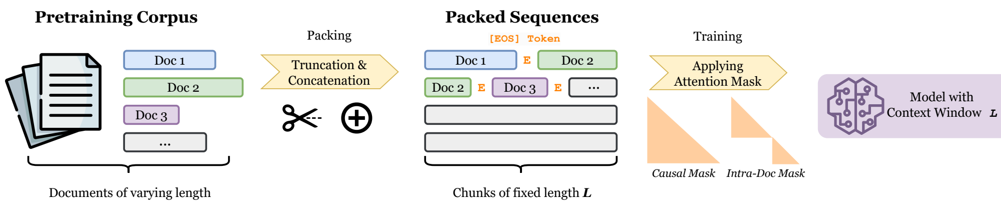


Figure 3: An illustration of the workflow for pretraining data preparation highlights several critical decisions. Key considerations include the method of data packing, the type of attention mask to employ (causal or intra-doc mask), and determining the appropriate context window length L.

others perform extended pretraining on longer-sequence corpora [12, 54, 32, 62]. Our approach differs from previous methods as we train *native* long-context models from scratch, rather than modifying a pretrained model in post-training. Compared with a naive long-context pretraining baseline with a constant schedule, our approach delivers substantial gains on multiple long-context tasks, underscoring the benefits of training from scratch. These findings show that our method can be a promising direction for future research on building language models with longer context windows.

3 How Context Window Affects Pretraining

How does context window affect pretraining? To investigate this in a fair and comparable manner, we pretrain language models from scratch with context windows ranging from 512 to 16,384 tokens under a fixed total number of tokens, evaluating via perplexity and downstream task benchmarks. We examine how the context window size impacts model performance, analyzing how data packing and masking strategies interact with window size.

3.1 Packing, Masking and Context Window

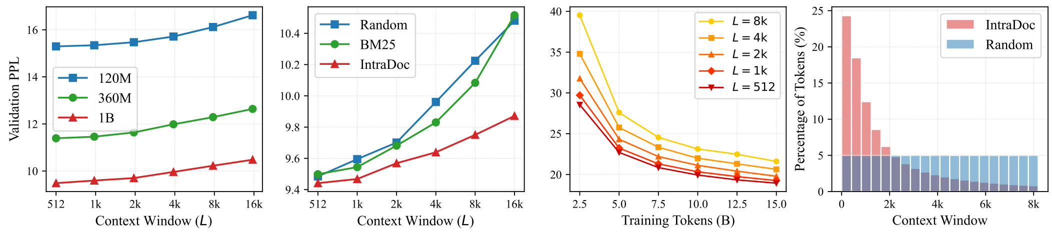
Most modern LLMs are based on a decoder-only transformer architecture [53] with a fixed context window size denoted by L. In contrast, the pretraining corpus, $D = \{d_1, d_2, d_3, \ldots, d_n\}$, consists of documents with varying lengths different from L. Therefore, a key step before pretraining is to *pack* the documents into sequences of length L. Formally, a packed sequence C_i is constructed as $C_i = \text{Trunc}(d_{i,1}) \oplus d_{i,2} \oplus \cdots \oplus d_{i,n-1} \oplus \text{Trunc}(d_{i,n})$, where $\oplus$ represents concatenation, and $\text{Trunc}(\cdot)$ denotes truncation of documents to ensure $\text{len}(C_i) = L$. Following previous works [44, 63], document boundaries within C_i are explicitly marked using end-of-sequence (`[EOS]`) tokens.

After the sequences are packed, the inputs are passed into transformer layers for next-token prediction training. A crucial component of these layers is the attention mechanism, which can be formulated as $A_{i,j} = q_i^\top k_j$, and then $\text{Attn}(X) = \text{Softmax}(A + M)$. In decoder-only models, a mask M is applied to introduce constraints. A common approach is to use a *causal mask*, which ensures that each position can only attend to previous tokens by masking out (setting to $-\infty$) attention scores corresponding to future positions: $M_{ij} = -\infty$ for $j > i$ and $M_{ij} = 0$ otherwise. A recently proposed masking scheme, known as *intra-doc* mask [63, 13], imposes a constraint that only allows tokens to attend to each other if they belong to the same document. Let each document d have start index s_d and end index e_d, the masking can be denoted as $M_{ij}^{\text{intra}} = 0$ when $\exists\, d$ such that $s_d \leq i, j \leq e_d$ and $j \leq i$, and $M_{ij}^{\text{intra}} = -\infty$ otherwise. The model is trained with the standard cross-entropy loss on the packed sequences of length L. The workflow for pretraining data processing is illustrated in Figure 3.

3.2 Preliminary Study on Context Window Size

As per Section 1, we initiate our study by investigating the impact of context window size on model performance through a controlled experiment. Specifically, we pretrain language models with varying context window sizes, while preserving all other experimental settings. This enables a pure analysis of the context window's influence on model performance. Through this analysis, we aim to understand whether *longer context windows inherently lead to better or worse model performance*.

Key Variables. The context window size determines the number of tokens included in the context for each packed sequence. However, as discussed earlier, several additional factors influence the content



(a) Validation perplexity of models with varying sizes across different context window sizes.

(b) Validation perplexity of different packing and masking strategies across varying context window sizes.

(c) Validation perplexity of models without positional encoding across varying context window sizes

(d) Distribution of context window sizes for IntraDoc and Random methods.

Figure 4: Ablation studies of different factors on different context window sizes. Note that the validation PPL is obtained on the validation documents with a sliding window size of 512 tokens. The packing strategy in (a) is Random, and the model sizes in (b) and (c) are 1B and 120M, respectively. Note that the context window in (d) means the number of available preceding tokens when making next-token prediction (calculation details in Section A.6).

within the context window: (1) *Packing methods* determine which documents constitute the context window, and different packing strategies can significantly alter the composition of token sequences; (2) *Masking methods* decide whether cross-document attention is enabled within the same context window. The choice of masking affects how the information from different documents interacts during training.

Packing and Masking. To study the impact of packing, we employ two strategies: *random packing* and *semantic packing*. For random packing, documents are randomly concatenated without a specific ordering. For semantic packing, inspired by Shi et al. [44], we retrieve and concatenate semantically relevant documents from the corpus, aiming to keep them within the same context window. After experimenting with both a dense retriever [20] and a lexical retriever BM25, we found that BM25 gives stronger performance and chose it as our focus. For masking, the baseline approach is causal masking, where each token can attend to all preceding tokens within the same context window, regardless of document boundaries. Conversely, recent studies [63, 9] show that disabling cross-document attention, thereby enabling intra-document attention, improves performance. For clarity in subsequent discussions, we denote random packing with causal masking as **Random**, BM25 packing with causal masking as **BM25**, and random packing with intra-document masking as **IntraDoc**.

Training. We pretrain models from scratch using the TinyLlama codebase [60], and study models with 120M, 360M and 1B parameters. Given the substantial computational cost associated with retrieval in semantic packing, we randomly select around 30B tokens from the CommonCrawl (CC) subset of the SlimPajama dataset [46] as the pretraining corpus. All models undergo training for up to 100B tokens ($\sim$3.3 epochs). To ensure consistency across experiments, we strictly control all other settings, retaining the same batch size and learning rate schedule for all context windows. All models also incorporate Rotary Positional Encoding (RoPE) [47] to encode positional information. Appendix A.3 and A.4 give further model architecture details and training settings.

Evaluation. For all model sizes, we use perplexity (PPL) on validation documents from the original dataset as a key metric, in line with established practices [10, 24, 17]. Note that when comparing models across different context windows (e.g., a 2K-context model and an 8K-context model), we must ensure the evaluation sequence fits within the shorter model's context window to maintain a fair comparison. We also evaluate 1B models on downstream standard benchmarks: HellaSwag [59], ARC-Easy and ARC-Challenge [6], Winogrande [42], CommonsenseQA [48], OpenBookQA [34], PIQA [2], Social-QA [43], and MMLU [16]. We employ the OLMES suite [15] for the evaluation, as it has been shown to provide reliable and stable results with curated 5-shot demonstrations [12].

3.3 Experimental Results

Figure 1 presents the main experimental result, obtained using the Random setting with 1B-parameter models. The results indicate that context window size significantly influences the performance of LLMs, with *shorter contexts generally leading to better performance*. To further investigate the factors contributing to the observation, we perform a comprehensive analysis to examine potential variables that may affect the conclusion. Figure 4 shows our results, and we derive four key findings:

Findings: (1) The advantage of training on shorter contexts is consistent across model sizes; (2) This advantage is independent of the packing and masking methods employed; (3) It is also unrelated to the use of positional encoding; (4) The best packing and masking strategy is IntraDoc, which outperforms others probably because it introduces a larger number of short contexts during pretraining.

Findings (1) and (2). As shown in Figure 4, regardless of the model size in (a) or the packing and masking methods in (b), a shorter context window for pretraining generally results in higher average performance on benchmarks. The finding on benchmarks is consistent with the trend of validation PPL, where shorter context windows always yield lower PPL.

Finding (3). When using shorter context windows, one might hypothesize that the model learns positional encoding patterns for nearer positions more frequently, leading to better performance on standard benchmarks. To test the hypothesis, we systematically ablate RoPE by completely excluding it during pretraining, following prior work [25]. In Figure 4(c), models trained with short-context windows still outperform their long-context counterparts, even in the absence of positional encoding. This suggests that the advantages of shorter contexts are independent of positional encoding.

Finding (4). From Figure 4(b), we observe that IntraDoc achieves the best validation PPL across all context window sizes compared to Random and BM25, alongside consistently higher performance on standard benchmarks (c.f. Appendix A.7.1). This raises the question: why does IntraDoc excel? We attribute the advantage to the context window size distribution of IntraDoc, which implicitly increases the prevalence of shorter contexts. As illustrated in Figure 4(d), despite the sequence length of 8K, fewer than 1% of context windows actually reach this limit. While prior work links the success of IntraDoc to reduced contextual noise [63], we identify a complementary factor — reduced average context window size — as a key factor in its strong performance. That is, we hypothesize that the effectiveness of IntraDoc may also be closely tied to short context windows.

4 SkyLadder: Context Window Scheduling

We now present SkyLadder for progressively expanding the context window during pretraining.

4.1 Method

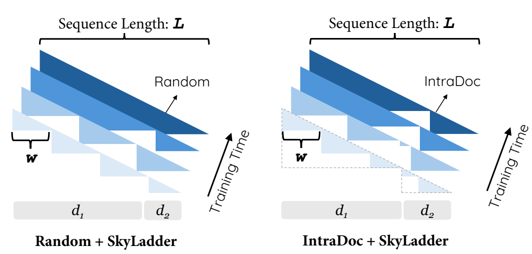


Figure 5: An illustration of SkyLadder with Random and IntraDoc. The example shows a packed sequence (length L) consisting of two documents. For SkyLadder, the context window w starts from a small value and dynamically adjusts during training, eventually converging to the masking patterns of Random or IntraDoc.

Inspired by learning rate scheduling, we explore whether dynamically scheduling the context window from short to long during pretraining could lead to performance improvements. This method can be implemented by applying multiple local "mini" causal masks to a long, packed sequence. We illustrate this masking strategy in Figure 5.

Formally, we define a local window length w. The associated mask M_w is defined as follows: $M_{ij} = 0$ when $\lfloor \frac{i}{w} \rfloor w \leq j \leq i$, and $M_{ij} = -\infty$ otherwise, where $\lfloor \frac{i}{w} \rfloor w$ calculates the largest multiple of w that is less than or equal to i, effectively defining a block-wise attention mask for the query token at position i. We linearly adjust the window size upwards by a constant factor per training step t: $w(t) = \min(w_e, w_s + \lfloor \alpha t \rfloor)$, where w_e and w_s represent the ending and starting context window sizes, respectively. Here, α denotes the rate of expansion, and t corresponds to the training step. As the training progresses, when the dynamic context window size $w(t)$ eventually reaches the desired (long) context window size $L = w_e$, it remains fixed at that value. At this point, the attention mask is equivalent to a full causal mask. Notably, this method modifies the effective context window through masking, independent of how the sequences are packed. As such, this mask M_w can be integrated with M^{Intra}, which maintains the attention boundaries between documents; it can be seamlessly combined with most packing and masking strategies.

Table 1: Performance (accuracy in %) of different 1B models pretrained on 100B CC tokens on standard benchmarks. * denotes statistical improvements over the baseline (described in §A.7.3).

Method	Avg.	ARC-E	ARC-C	CSQA	HS	OBQA	PIQA	SIQA	WG	MMLU
Random	46.3	58.0	32.7	49.6	43.0	40.2	64.8	46.4	51.9	29.9
+ SkyLadder	**50.0** (+3.7)	**65.4***	**35.6***	**56.8***	**47.0***	**42.8**	64.8	**48.9***	56.0*	**32.4***
IntraDoc	47.4	61.8	33.4	52.7	45.6	38.0	64.3	45.7	54.8	30.5
+ SkyLadder	49.3 (+1.9)	64.8*	33.8	55.4*	**47.9***	39.4	**66.1***	48.0*	**56.4**	31.8*

Table 2: Performance (accuracy in %) of 1B models pretrained on 100B CC tokens with different methods on reading comprehension and long-context benchmarks. Detailed setup is in Appendix A.7.3.

Method	Reading Comprehension Benchmarks						Long Benchmarks		
	Avg.	HotpotQA	SQuAD	NQ	TriviaQA	RACE-h	Avg.	MDQA	RULER
Random	25.5	6.5	37.0	15.8	37.7	30.7	**15.3**	17.7	**12.8**
+ SkyLadder	**30.2** (+4.7)	**12.4**	**40.2**	**20.4**	**43.0**	**35.0**	14.3	**18.3**	10.3
IntraDoc	28.7	11.4	39.0	18.2	42.3	32.3	13.0	15.3	10.6
+ SkyLadder	29.1 (+0.4)	11.0	38.5	**20.4**	41.5	34.3	13.2	15.6	10.7

4.2 Experimental Setup

We follow the same setup in Section 3.2 to pretrain language models with 8K context on 100B tokens. We set $w_s = 32$ and $\alpha = 1/8$ by default, which means that a model roughly needs 64K steps (around 64B tokens) to reach the final desired context window of $L = 8192$. All baseline and SkyLadder models are implemented with Flash Attention 2 [7] (pseudocode in A.5). We fix all other hyperparameters, such as the learning rate schedule, batch size, etc., for fair comparison. Due to resource constraints, we do not perform extensive hyperparameter search to obtain the best combinations for $w(t)$, α, and w_s. In our ablation study, we show that these hyperparameters have a negligible impact on performance, as long as they are within a reasonable range.

For evaluation, we use the same suite mentioned in Section 3.2 with standard benchmarks. To evaluate the performance of long-context question answering within an 8K length, we utilize the 30-document setting from the Multi-Document QA (MDQA) benchmark [30]. This is a widely-adopted benchmark that is shown to be reliable for models of 1B scale [38, 63], with an average length of approximately 6K tokens. We also select synthetic tasks within RULER [18], as defined by Yen et al. [58]. We choose the setup of the task that fills up the model's target context window L.

4.3 Experimental Results

Tables 1 and 2 present the main results, highlighting significant improvements achieved by SkyLadder across standard benchmarks, reading comprehension tasks and long-context benchmarks. For instance, compared to the Random baseline, integrating SkyLadder yields notable performance gains on standard tasks such as MMLU (+2.5%), ARC-E (+7.4%), and HellaSwag (+4%). This suggests that models with SkyLadder excel at learning common knowledge during pretraining. Additionally, our method further improves the performance of the strong baseline IntraDoc across many benchmarks. Meanwhile, for realistic long-context benchmarks like MDQA, SkyLadder matches or exceeds baseline performance. For RULER, the performance difference is likely because of fluctuation caused by its synthetic nature and small size [54]. More long-context evaluation can be found in Section A.7.3, confirming that SkyLadder is comparable with or better than baselines on long-context evaluation. In addition to Random and IntraDoc, we also verify that SkyLadder improves the performance of the BM25 model on both short and long tasks (Section A.7.4).

To address potential concerns that the benefits observed in short contexts may stem from the high level of noise in CC, we conduct additional experiments using the FineWeb-Pro dataset [64], a carefully curated high-quality dataset containing 100B tokens. As shown in Table 4, improved data quality indeed leads to substantial performance gains. However, our key findings remain consistent: IntraDoc continues to outperform Random, and SkyLadder consistently delivers significant improvements over both baselines. This demonstrates that our method generalizes to corpora of varying quality.

We further examine whether SkyLadder is generalizable beyond natural language tasks. Following Ding et al. [9], we pretrain 1B code models on 100B Python code with the Starcoder tokenizer [29].

Table 3: Performance (in %) of 1B models pretrained on 100B Python code data. We follow the protocol of Huang et al. [19] to evaluate on HumanEval [3] and BigCodeBench [65]. t is the sampling temperature. SkyLadder shows consistent improvement especially for 32K-context models.

		HumanEval			**BigCodeBench**		
		Greedy	Sampling ($t = 0.8$)		Greedy	Sampling ($t = 0.8$)	
L	Method	Pass@1	Pass@10	Pass@100	Pass@1	Pass@10	Pass@20
32K	Random	17.7	32.4	51.8	9.0	16.1	19.7
	+ SkyLadder	**21.3**	**37.7**	**59.8**	**9.4**	**20.6**	**24.3**
8K	Random	22.0	37.2	61.0	9.9	19.3	23.6
	+ SkyLadder	**23.2**	**38.2**	**63.4**	**11.3**	**20.0**	**24.1**

Table 4: Performance (average accuracy over tasks, in %) of 1B models pretrained on FineWeb-Pro with an 8K context window.

Method	**Standard**	**Long**
Random	52.5	11.1
+ SkyLadder	**55.2** (+2.7)	12.3 (+1.2)
IntraDoc	54.3	12.7
+ SkyLadder	54.8 (+0.5)	**13.9** (+1.2)

Table 5: Performance (average accuracy in %) for models of different sizes.

Size	**Method**	**Standard**	**Long**
120M	Random	40.1	5.8
	+ SkyLadder	41.2 (+1.1)	5.1 (-0.7)
360M	Random	47.2	8.9
	+ SkyLadder	49.6 (+2.4)	8.9
3B	Random	57.0	15.8
	+ SkyLadder	**60.5** (+3.5)	**19.3** (+3.5)

We observe a lower training loss (~ 0.9) for code pretraining compared to natural language (~ 2.1), suggesting that the structure in code makes the training easier. However, as shown in Table 3, there is still significant improvement when applying SkyLadder under both greedy decoding and sampling setups, especially when the target context length is 32K. This demonstrates the potential of SkyLadder to coding and possibly other reasoning tasks beyond natural language modelling.

4.4 Scalability Experiments

We examine whether SkyLadder's improvements persist as we scale up the model parameters and extend the context window size. We use the largest model and context size that our compute permits.

Model Size. We conduct experiments across three model sizes: 120M, 360M, and 3B parameters on the Fineweb-Pro dataset. Table 5 demonstrates that models utilizing SkyLadder consistently achieve better standard benchmark performance on all model sizes. For long context tasks, our method does not benefit 120M models, possibly due to their limited capacity in processing long sequences. However, the performance gain on 3B models is prominent. We observe a positive scaling trend: as the model size grows, the performance improvement also increases, indicating the potential of applying our method to even larger models beyond our current scale. We leave it as a future work to explore larger models as it requires significantly more compute.

Context Window Size. To examine whether SkyLadder can effectively scale to longer context windows, we train 1B models with a 32K context window on 100B FineWeb-Pro tokens. We adjust α to $1/2$ to ensure that the final context window expands to 32K before the end of pretraining. As shown in Table 6, our model demonstrates strong performance on both standard and long benchmarks. In addition, the performance difference of SkyLadder (0.9%) between the 8K and 32K models is largely reduced compared with the baseline approach (1.8%), which alleviates the performance degradation described in our earlier study. Notably, compared to the baseline Random approach, SkyLadder trains the model on progressively shorter contexts during earlier stages. This reveals a counterintuitive insight: naively training a model with a long context window is not always optimal, even if the model is evaluated on long contexts. In contrast, strategic scheduling of the context window during pretraining can yield better results.

Table 6: Performance (%) of 1B models trained on 100B FineWeb-Pro tokens with a 32K context window.

Method	**Standard**	**Long**
Random	50.7	9.7
+ SkyLadder	54.3 (+3.6)	13.5 (+3.8)
IntraDoc	54.0	13.0
+ SkyLadder	**54.9** (+0.9)	**14.4** (+1.4)

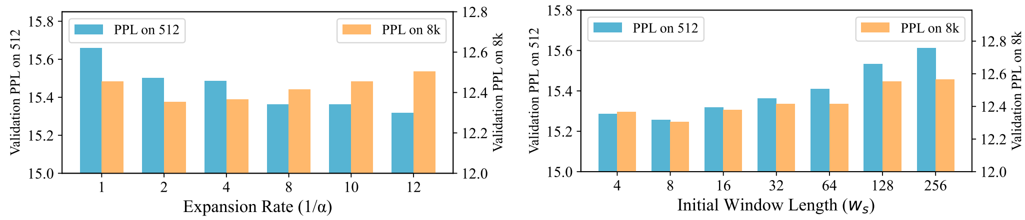


Figure 6: Validation PPL on 512 and 8K contexts of models with different expansion rate α (left) and initial window length w_s (right).

Table 7: Comparison of 1B models trained with a 32K context window with different scheduling methods. Numbers are average accuracy (%).

Method	**Long**	**Standard**
Constant Long (32K)	9.7	50.7
Linear (32→32K, *default*)	13.5	54.3
Stepwise Linear (32→32K)	13.3	**55.3**
Sinusoidal (32→32K)	**14.2**	54.2
Exponential (32→32K)	11.5	54.7
Cont. Pretrain (4K→32K)	2.7	52.9

Table 8: Comparison of relative training time and compute efficiency for 1B Models with different context window sizes L. FLOPs calculation follows Zhang et al. [60]. A larger context window leads to more efficiency gains.

Method	**Time** (%)	**FLOPs** (10^{20})
Random (8K)	100.0%	11.6
+ SkyLadder	86.9% (-13.1%)	9.9 (-14.7%)
Random (32K)	100.0%	25.5
+ SkyLadder	77.8% (-22.2%)	18.8 (-26.3%)

4.5 Ablation Study

We now examine the impact of hyperparameters in SkyLadder scheduling. To manage computational costs, we adopt a default setup of pretraining 120M models with 8K context on 100B CC tokens.

Expansion Rate. We investigate the impact of the expansion rate α in Figure 6 (left). We choose different α ranging from slowest ($1/12$) to fastest (1). Our findings reveal that, for short contexts, performance generally improves as the expansion rate slows down. However, selecting an excessively slow rate (e.g., $1/12$) can negatively affect long-context performance due to insufficient training on longer contexts. **Therefore, we recommend setting α to $1/8$ for a good balance.**

Initial Context Window. As the final context window length w_e is fixed to L, the sole remaining hyperparameter is w_s. Intuitively, setting w_s to an excessively large value (e.g. close to L) leaves little room for scheduling, resulting in sub-optimal performance. In Figure 6 (right), we demonstrate that when w_s is set to a relatively small value (e.g., 8), great performance can be achieved for both short and long contexts. This suggests that there is still potential for further improvement in our default setup. **Therefore, we recommend starting with a small context window, such as 8 tokens**.

Scheduling Type. The default scheduling method in SkyLadder is linear scheduling. We evaluate different context window scheduling types (more details in Table 18 and Figure 11 in Appendix A.7.4): (1) **Stepwise Linear** rounds window size $w(t)$ to multiples of 1K, resulting in a step function; (2) **Sinusoidal** increases quickly at the early stage then slows down; (3) **Exponential** starts slow but accelerates sharply; (4) **Continual pretraining** setup trains with 4K context windows for $\sim$97B tokens, then switches to 32K context for the final 3B tokens. Table 7 shows that linear and sinusoidal schedules outperform the exponential variant on long tasks, likely because the exponential schedule, with extended short-context pretraining at the beginning, fails to adequately train on long contexts. Last, the most commonly used continual pretraining setup performs poorly overall, suggesting abrupt context changes harm both short and long performance. **These findings suggest that context window scheduling is superior to both constant long-context pretraining and continual pretraining**.

Overall, we conclude that the schedule should start from a small w_s and the expansion should be gradual. We leave it to future work to study more advanced schedules and discover optimal configurations. For instance, it is possible that the schedule needs to be adjusted for various model sizes. More ablations for combination with BM25, hybrid attention, cyclic schedules and scheduling under a compute budget can be found in Appendix A.7.4.

4.6 Analysis and Discussion

Training Efficiency. We observe a significant boost in training efficiency when employing SkyLadder in Table 8. On 8K models, SkyLadder accelerates training time by 13% due to the reduced context window in calculating attention. With a 32K context window, the efficiency gain becomes even more pronounced: our method saves 22% of training time while achieving better performance. The FLOPs saving is larger than the actual time because of reduced attention calculation.

Attention Pattern. We next investigate why SkyLadder, despite being trained on short contexts overall, consistently outperforms the baseline. As language models rely on attention mechanisms to encode context information, we study how attention patterns change. Specifically, during pretraining, we monitor the dynamics of (i) attention entropy (solid lines in Figure 7), where a lower entropy is associated with better downstream performance [61]; (ii) attention sink [55], where the initial token in the context receives disproportionately high attention. We utilize the metric in Gu et al. [14] to quantitatively measure the amplitude of attention sink. As shown in Figure 7 (dashed lines), compared with the baseline Random, SkyLadder demonstrates reduced attention entropy, suggesting a more concentrated attention pattern. However, a slower emergence and lower amplitude of attention sink are simultaneously observed. This suggests that SkyLadder's attention is concentrated on the key information in the context instead of the initial token, which accounts for the performance gain.

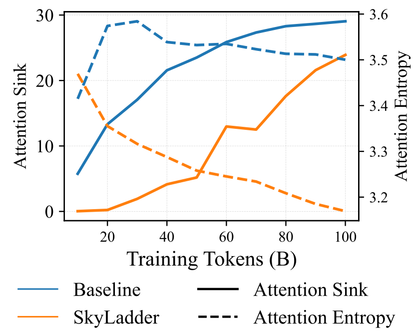


Figure 7: Dynamics of attention sink and entropy during pretraining 1B models (8K context). SkyLadder delays the emergence of attention sink while lowering the overall entropy, indicating a more effective attention pattern.

Comparison with Related Work We compare our method with another approach for improving pretraining in Table 9. As discussed in Section 2, Pouransari et al. [38] proposed Dataset Decomposition (DD) by segmenting a document into sequences of varying lengths and using a curriculum during pretraining. However, this approach inevitably introduces domain bias, as the document lengths in different domains are different [10]. This explains why DD with only one short-to-long cycle fails to outperform the IntraDoc baseline. To mitigate this, the authors suggested iterating through multiple cycles of long and short data, which does improve performance substantially. In contrast, our method achieves better performance by avoiding such biases by not altering the data order based on length. In Appendix A.7.4, we experimented with various cyclic schedules but did not observe any improvements. In fact, we noticed loss spikes between cycles (Figure 13), indicating potential issues with domain shifts. This further supports that our method is safer since it does not disrupt the natural ordering and distribution of the data. More discussion with other related works [28, 21] is in Section A.8, where we demonstrate that our work provides novel insights that scheduling the context window over the entire training time improves both efficiency and performance.

Table 9: Comparison between SkyLadder and Dataset Decomposition (DD) on 1B models trained with 100B FineWeb-Pro tokens. Numbers are in average performance in %.

Model	Standard	Long
IntraDoc	54.3	12.7
+ SkyLadder	**54.8** (+0.5)	**13.9** (+1.2)
+ DD (1 cycle)	53.9 (-0.4)	12.3 (-0.4)
+ DD (8 cycles)	54.5 (+0.2)	13.5 (+0.8)

5 Conclusion

We conduct a comprehensive controlled study of the impact of context window on pretraining, revealing that a shorter context window is more beneficial to the model's performance on standard benchmarks. This debunks the trend of pretraining with longer context windows. We therefore propose SkyLadder to schedule the context window from short to long during pretraining, which gives substantial improvement in downstream performance and computational efficiency. We conclude that context window scheduling is an important dimension for pretraining, and deserves more consideration. In the future, we plan to explore more dynamic and performant scheduling strategies that adapt according to model size or pretraining data distribution.

References

[1] Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short?, 2024. URL https://arxiv.org/abs/2410.18745.

[2] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

[4] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023. URL https://arxiv.org/abs/2306.15595.

[5] Yu-An Chung, Hung-Yi Lee, and James Glass. Supervised and unsupervised transfer learning for question answering, 2018. URL https://arxiv.org/abs/1711.05345.

[6] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

[7] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

[9] Hantian Ding, Zijian Wang, Giovanni Paolini, Varun Kumar, Anoop Deoras, Dan Roth, and Stefano Soatto. Fewer truncations improve language modeling. In *Proceedings of the 41st International Conference on Machine Learning*, pages 11030–11048, 2024.

[10] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In *Proceedings of the 41st International Conference on Machine Learning*, pages 14125–14134, 2024.

[11] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

[12] Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively), 2025. URL https://arxiv.org/abs/2410.02660.

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[14] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view, 2024. URL https://arxiv.org/abs/2410.10781.

[15] Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5005–5033, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL https://aclanthology.org/2025.findings-naacl.282/.

[16] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[17] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[18] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

[19] Siming Huang, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for top-tier code large language models, 2024. URL https://arxiv.org/abs/2411.04905.

[20] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning, 2021. URL https://arxiv.org/abs/2112.09118.

[21] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Chia-Yuan Chang, and Xia Hu. Growlength: Accelerating LLMs pretraining by progressively growing training length, 2023. URL https://arxiv.org/abs/2310.00576.

[22] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. LLM maybe longlm: Selfextend LLM context window without tuning. In *Forty-first International Conference on Machine Learning*, 2024.

[23] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

[24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[25] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[26] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

[27] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082/.

[28] Conglong Li, Minjia Zhang, and Yuxiong He. The stability-efficiency dilemma: investigating sequence length warmup for training gpt models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 26736–26750, 2022.

[29] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023. URL https://arxiv.org/abs/2305.06161.

[30] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

[31] LocalLLaMA. NTK-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degration, 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

[32] Yi Lu, Jing Nathan Yan, Songlin Yang, Justin T. Chiu, Siyu Ren, Fei Yuan, Wenting Zhao, Zhiyong Wu, and Alexander M. Rush. A controlled study on long context extension and generalization in llms, 2024. URL https://arxiv.org/abs/2409.12181.

[33] Xin Men, Mingyu Xu, Bingning Wang, Qingyu Zhang, Hongyu Lin, Xianpei Han, and Weipeng Chen. Base of RoPE bounds context length, 2024. URL https://arxiv.org/abs/2405.14591.

[34] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

[35] Koichi Nagatsuka, Clifford Broni-Bediako, and Masayasu Atsumi. Pre-training a BERT with curriculum learning by increasing block-size of input text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 989–996, 2021.

[36] Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. QuALITY: Question answering with long input texts, yes! In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391/.

[37] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

[38] Hadi Pouransari, Chun-Liang Li, Jen-Hao Chang, Pavan Kumar Anasosalu Vasu, Cem Koc, Vaishaal Shankar, and Oncel Tuzel. Dataset decomposition: Faster llm training with variable sequence length curriculum. *Advances in Neural Information Processing Systems*, 37:36121–36147, 2024.

[39] Alec Radford. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

[40] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

[42] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[43] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

[44] Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Xi Victoria Lin, Noah A Smith, Luke Zettlemoyer, Wen-tau Yih, and Mike Lewis. In-context pretraining: Language modeling beyond document boundaries. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[46] Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 6 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

[47] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[48] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the*

2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

[49] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

[50] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[51] 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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

[52] Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. Towards machine comprehension of spoken content: Initial toefl listening comprehension test by machine, 2016. URL https://arxiv.org/abs/1608.06378.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[54] Haonan Wang, Qian Liu, Chao Du, Tongyao Zhu, Cunxiao Du, Kenji Kawaguchi, and Tianyu Pang. When precision meets position: Bfloat16 breaks down rope in long-context training, 2024. URL https://arxiv.org/abs/2411.13476.

[55] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

[56] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oğuz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models. In *North American Chapter of the Association for Computational Linguistics*, 2023.

[57] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

[58] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly. In *International Conference on Learning Representations (ICLR)*, 2025.

[59] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[60] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024. URL https://arxiv.org/abs/2401.02385.

[61] Zhisong Zhang, Yan Wang, Xinting Huang, Tianqing Fang, Hongming Zhang, Chenlong Deng, Shuaiyi Li, and Dong Yu. Attention entropy is a key factor: An analysis of parallel context encoding with full-attention-based pre-trained language models, 2024. URL https://arxiv.org/abs/2412.16545.

[62] Liang Zhao, Tianwen Wei, Liang Zeng, Cheng Cheng, Liu Yang, Peng Cheng, Lijie Wang, Chenxia Li, Xuejie Wu, Bo Zhu, Yimeng Gan, Rui Hu, Shuicheng Yan, Han Fang, and Yahui Zhou. Longskywork: A training recipe for efficiently extending context length in large language models, 2024. URL https://arxiv.org/abs/2406.00605.

[63] Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. Analysing the impact of sequence composition on language model pre-training. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7897–7912, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.427. URL https://aclanthology.org/2024.acl-long.427/.

[64] Fan Zhou, Zengzhi Wang, Qian Liu, Junlong Li, and Pengfei Liu. Programming every example: Lifting pre-training data quality like experts at scale, 2024. URL https://arxiv.org/abs/2409.17115.

[65] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions, 2024. URL https://arxiv.org/abs/2406.15877.

Table 10: Model configurations for pretrained language models.

Model	`Tinyllama 1B`	`Tinyllama 120M`	`Tinyllama 360M`	`Llama3.2 3B`
Vocab Size	32000	32000	32000	32000
Layers	22	12	18	28
Heads	32	12	16	24
Embedding Dim	2048	768	1024	3072
Intermediate Size	5632	2048	4096	8192
Normalization	RMSNorm	RMSNorm	RMSNorm	RMSNorm
Normalization ϵ	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Query Groups	4	1	16	8
Bias	No	No	No	No
RoPE θ	10000 if $L = 8K$ 1000000 if $L = 32K$	10000	10000	100000

A Appendix

A.1 Limitations

While we perform extensive experiments to study the impact of context window on pretraining and demonstrate the effectiveness of SkyLadder, we acknowledge that there are still limitations to be addressed. First, we conduct experiments up to a 3B-model scale and 32K context length, while the latest large language models are typically much larger and capable of processing longer contexts. However, pretraining a large model with a long context window requires prohibitive computational resources beyond our budget. Within our computational capabilities, we have tried to demonstrate the generalizability of SkyLadder across corpora, context window size, model size, and downstream tasks. Thus, we leave it as future work to apply SkyLadder to larger models. Second, we do not include a theoretical analysis to explain the effectiveness of SkyLadder as we mainly focus on empirical insights. We suggest that future work may investigate the relationship between the context window and the training compute to obtain the optimal context window schedule.

A.2 Broader Impacts

The work aims to investigate the impact of choices on context windows in language model pretraining and proposes a way to speed up pretraining by scheduling context windows. On the positive side, this improves the efficiency of language model pretraining, making it more accessible and reducing the carbon footprint. Moreover, it enhances the performance of pretrained language models, which may result in better downstream performance in applications. There might be potential misuse of pretrained language models, which is beyond the scope of this work.

A.3 Model Architecture

In Table 10, we list the architecture choices of the models trained, including the 120M, 360M, and 1B models based on the TinyLlama architecture [60]. The 3B model is based on Llama3.2 architecture [13].

A.4 Training Configurations

We include details of the training configurations in Table 11. All models, irrespective of size or context window length, are trained on this same set of hyperparameters. For most of the hyperparameter values, we follow the TinyLlama [60] project, therefore, our results are highly reproducible.

A.5 Implementation

We provide the pseudocode for implementing SkyLadder with Flash Attention 2 [7]. The only change is to apply local causal masking with size w, and combine them with the original document boundaries under the IntraDoc scenario. It can easily be integrated into any model before calculating attention. The rest of the training pipeline remains unchanged.

Table 11: Hyperparameters setup for pretraining the language models. All pretrained models follow the same structure.

Parameter	Value
Optimizer	AdamW
AdamW-β_1	0.9
AdamW-β_2	0.95
Learning Rate Schedule	Cosine
Peak Learning Rate	4e-4
Minimum Learning Rate	4e-5
Warmup Steps	2000
Gradient Norm Clipping	1
Total Steps	100,000
Global Batch Size	1,048,576 (2^{20}) tokens
Weight Decay	0.1

SkyLadder with Flash Attention 2

```
# q, k, v: RoPE-encoded query, key, value tensors
# doc_boundaries: EOS token positions per document
# is_intradoc: intra-document attention flag
# training_step: current global step
# L: maximum context window length

# get current window size
w = min(L, get_current_mask_length(training_step))

# breakpoints every w tokens (and at document boundaries if using IntraDoc
    masking)
mask_boundaries = np.arange(w, L, w)
if is_intradoc:
    mask_boundaries = np.union1d(mask_boundaries, doc_boundaries)

# compute max segment length & cumulative lengths for flash attention
max_seqlen = get_max_seqlen(mask_boundaries, L)
cu_seqlens = get_cu_seqlens(mask_boundaries, L)

attn = flash_attn_varlen_func(
    q, k, v,
    cu_seqlens,
    max_seqlen,
    causal=True
)
```

A.6 Definition of Per-token Context Window

In Figure 4(d), we show the context window distribution difference between IntraDoc and Random. To clarify, the context window size refers to the number of preceding tokens available in the context window when making the next token prediction. This is different from (a) and (b), where the context length L is the model's pretrained context window.

Formally, consider a token at index i and an attention mask matrix M, where an entry $M_{i,j} = 0$ indicates that token i can attend to token j, and $-\infty$ otherwise. The context window size C_i for the i-th token is defined as $C_i = \sum_{j=1}^{i} \mathbf{1}\left\{M_{i,j} = 0\right\}$, where $\mathbf{1}\left\{\cdot\right\}$ is the indicator function that returns 1 when $M_{i,j} = 0$ and 0 otherwise. In essence, C_i is the number of tokens available as context for the i-th token, and the distribution of C_i over all pretraining tokens is in Figure 4(d).

For Random, the causal mask is triangular: the i-th token has a context window size equal to i (i.e., $C_1 = 1$, $C_2 = 2$, etc.). Thus, the distribution of C_i is uniform. In contrast, IntraDoc effectively shortens the context length by limiting the cross-document attention.

A.7 Additional Results

A.7.1 Context Window Study

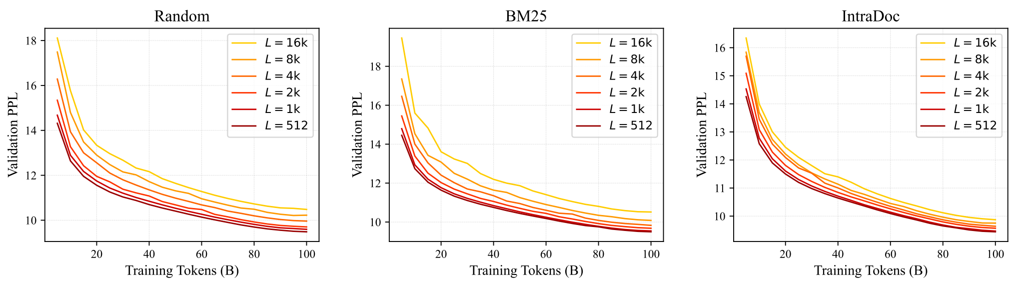


Figure 8: Validation perplexity (evaluated on a sliding window of 512) on models with different context lengths.

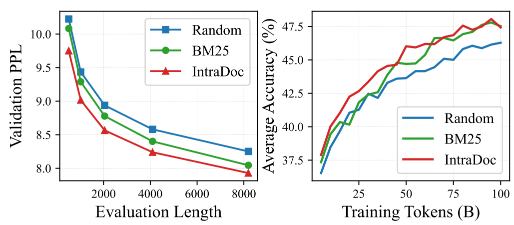


Figure 9: Left: Evaluation perplexity of models with different packing or masking strategies. Right: Downstream performance over 9 tasks of different models.

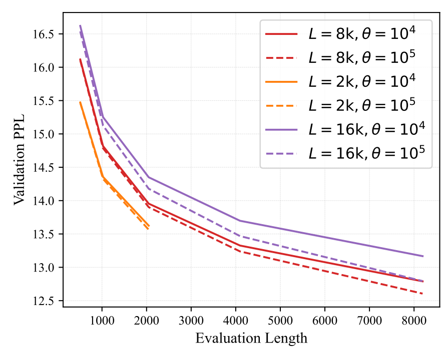


Figure 10: Validation perplexity vs training tokens with different context windows and base of RoPE, θ. Evaluation is done on a sliding window of varying length (x-axis) on the validation documents.

In Figure 8. We plot the validation perplexity of models with different context windows under the Random, IntraDoc, and BM25 settings. We observe a consistent trend that a shorter-context model has lower evaluation perplexity on a shorter sequence under all settings.

In Figure 9, we plot the evaluation perplexity and downstream performance of models with different packing or masking strategies. We conclude that overall, IntraDoc achieves the best performance, with a consistently lower PPL and a higher downstream accuracy. We think that this is partially due to the shorter context window that the IntraDoc model is trained on.

A.7.2 Ablations for Context Window Study

Base of RoPE. It has been shown that the value of RoPE may have a significant impact on the model's long context performance, and a longer context requires a larger base [33]. Therefore, we increase the RoPE base to 100,000, which is sufficiently large according to Men et al. [33]. In Figure 10, we observe an improvement for long-context models on long-context evaluation. However, the large gap between a shorter and a longer model still remains, therefore rejecting the hypothesis that the RoPE base is the key contributing factor to the superior performance of short-context models.

Table 12: Performance of 3B models on long tasks of retrieval-augmented generation (evaluated by exact-match scores) and reading comprehension benchmarks (accuracy in %).

	Retrieval Augmented Generation					Reading Comprehension		
Model	Avg.	NQ	TriviaQA	HotpotQA	PopQA	Avg.	TOEFL	QuALITY
Random	30.3	24.3	45.2	29.3	22.5	37.1	43.5	30.6
+ SkyLadder	**35.5**	**27.8**	**52.7**	**32.3**	**29.3**	**39.4**	**48.0**	**30.9**

Table 13: Many-shot ICL performance (accuracy) on text classification benchmarks. Numbers in parentheses denote the number of labels for each task.

Model	**Avg.**	**DBpedia (14)**	**AGNews (4)**	**Amazon (2)**	**Yelp (2)**	**SST2 (2)**
Random	73.9	17.4	68.6	**94.3**	94.7	**94.5**
+ SkyLadder	**76.5**	**25.5**	**75.8**	94.1	**95.0**	92.2

A.7.3 SkyLadder Evaluation

Statistical Test We test the statistical significance of the performance difference between our models and baselines in Table 1. We use a McNemar test as the two models are evaluated on the same set of questions. The original OLMES suite samples 1000 examples from each benchmark's full evaluation suite. In contrast, when conducting the McNemar test, we evaluate models on the full set to obtain more statistically meaningful results. We note that OpenBookQA only has 500 questions, making it harder to obtain statistical significance.

Reading Comprehension For reading comprehension, we evaluate the following benchmarks: Hotpot QA (2-shot) [57], SQuAD (4-shot) [41], NaturalQuestions (NQ) (2-shot) [26], TriviaQA (2-shot) [23], and RACE-high (0-shot) [27]. We follow the setup by Zhao et al. [63], where NQ and TriviaQA use retrieved documents as contexts. For RACE, we use `lm-evaluation-harness` [11] to compare the PPL between options.

Long-context Evaluation We provide additional long-context evaluation on our largest 3B model with an 8K context. This is to mitigate the performance instability of using synthetic benchmarks on small models. We first follow [38] to evaluate model accuracy on reading comprehension benchmarks TOEFL [5, 52] and QuALITY [36]. Next, we evaluate the model's performance on Retrieval Augmented Generation (RAG), where the model is provided with many relevant but potentially noisy contexts and needs to locate the correct information. As shown in Table 12, SkyLadder consistently performs better than the baseline across all evaluated RAG and reading comprehension datasets, highlighting its ability to locate correct answers within a lengthy context. In addition, we test the in-context learning ability of the models on text classification benchmarks [63, 44]. Results in Table 13 suggest that SkyLadder shows a significant gain for tasks with many labels, such as DBpedia, while achieving comparable high performance on binary tasks.

Closed-book QA We additionally evaluate the closed-book QA performance of our models without access to any document. We use the evaluation protocol Zhao et al. [63] to measure the exact match. In Table 14, we notice a significant improvement in our methods compared to the baselines for answering closed-book questions. This is consistent with the results that our models show improvements on standard benchmarks that contain commonsense knowledge.

Table 14: 1B model (trained on CC) performance (exact match %) on closed-book QA tasks.

	Closed-book QA		
Model	NQ	TriviaQA	Average
Random	6.1	11.9	9.0
+ SkyLadder	9.0	17.5	**13.2**
IntraDoc	7.8	14.7	11.3
+ SkyLadder	8.2	17.4	**12.8**

A.7.4 SkyLadder Ablations

Combination with BM25 Packing As SkyLadder only changes the context length via masking without altering the underlying data, it is orthogonal to any advanced data packing method such as Shi

Table 15: Performance (%) of 1B models with different schedule types. All models are trained on the same 100B CommonCrawl tokens with a final context length of 8K. BM25 packing, when combined with SkyLadder, significantly boosts performance on long tasks.

	Standard Avg.	Long Avg.
Random	46.3	15.3
BM25	47.5 (+1.2)	16.4 (+1.1)
+ SkyLadder	**49.8** (+3.5)	**17.0** (+1.7)

Table 16: Performance (%) of 1B models with different schedule types. All models are trained on the same 100B FineWeb-Pro tokens with a final context length of 8K. Short-to-long scheduling is consistently better than long-to-short scheduling.

	Standard Avg.	Long Avg.
No Scheduling	52.5	11.1
Short-to-Long	**55.2** (+2.7)	**12.3** (+1.2)
Long-to-Short	52.6 (+0.1)	10.7 (-0.4)

Table 17: Evaluation perplexity for models with different evaluation context lengths L_e. All models are trained with 100B tokens on CommonCrawl.

Model	$L_e = 512$	$L_e = 4K$	$L_e = 8K$
Random – 120M	15.9	13.4	13.0
+ SkyLadder	15.5	12.9	12.4
Random – 360M	12.1	10.2	9.8
+ SkyLadder	**11.6**	**9.7**	**9.4**

et al. [44], Ding et al. [9]. In Table 15, we combine the SkyLadder with the BM25 packing method. We show that the model achieves even better performance on both short and long context evaluation than BM25 without scheduling, which is also better than the Random baseline. This reveals that our method can be combined with more advanced packing techniques to further boost performance.

Combination with Hybrid Attention We note that a recent interesting trend in pretraining models with long context is to use a hybrid attention structure. For instance, Gemma3 [49] uses a mixture of global and local attention layers to balance efficiency and performance of the long-context model. We are curious about the generalizability of SkyLadder to such architecture, and follow Gemma3's strategy with a global-to-local ratio of 6:1. The results are presented in Table 17. We observe that SkyLadder consistently outperforms the baseline across all evaluation lengths, verifying its applicability. Importantly, SkyLadder works along the time dimension and is combinable with different attention variants, as long as there is a context window to be scheduled.

Long-to-Short Schedule A possibility that SkyLadder works better than baseline on standard benchmarks, which are typically short, might be that the training data mix has more short-context data after applying the mask. To study the effect of pure data distribution, we conduct an ablation of reversing the original short-to-long schedule and name it as the long-to-short schedule. This schedule spends the same number of tokens (64B) in the changing phase, before the constant training phase in $L = 8K$ for another 36B tokens. In Table 16, we show that the long-to-short schedule is not helpful to the model's performance in both short and long evaluation tasks. This highlights that the context window needs to be scheduled, rather than simply having a data mixture of long and short contexts.

Alternative Schedule Types We explore various types of short-to-long scheduling following different functions as mentioned in Section 4.5. Table 18 shows the details of the schedule as a function of t, and Figure 11 shows an illustration of the different schedule types. In Table 7, we show that a smoother increase following the sinusoidal schedule works the best for long-context evaluation, while also achieving strong performance on standard benchmarks.

Expansion Rate We illustrate the effect of the rate of expansion α in Figure 12. As the evaluation is done on 8K contexts, models with a lower rate (and shorter context window) will have a higher loss as the evaluation length is out-of-distribution. However, eventually, all models' loss converges to a low level after the schedule reaches 8K. The detailed numbers of validation loss after pretraining can be found in Table 19. Following previous work [10, 17, 24], we consider a loss difference larger than 0.01 as significant. We conclude that setting a reasonable rate of $1/8$ balances both short and long-context loss, which is the default setup for our main experiments.

Table 18: Functions for different context window schedule types. We set $w_s = 32$ and $w_e = 32768$ in our experiments. The r for rounding is set to 1024.

Schedule	Function
Constant	w_e
Linear	$w_s + (w_e - w_s)\frac{\alpha x}{w_e - w_s}$
Stepwise	$\max(w_s, r \times \left\lfloor \frac{L(x)}{r} \right\rfloor)$
Sinusoidal	$w_s + (w_e - w_s)\sin\left(\frac{\alpha \pi x}{2(w_e - w_s)}\right)$
Exponential	$w_s \times \left(\frac{w_e}{w_s}\right)^{\frac{\alpha x}{w_e - w_s}}$

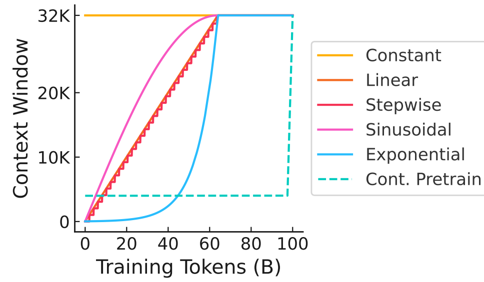


Figure 11: Illustration plot of various scheduling types.

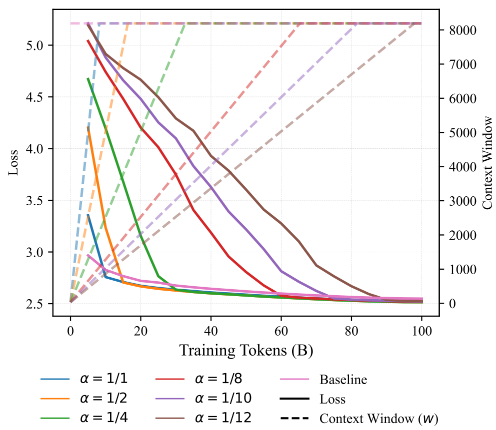


Figure 12: An illustration of the effect of different α. Dashed lines represent the current context window w for each step, and solid lines are the loss evaluated at 8K length.

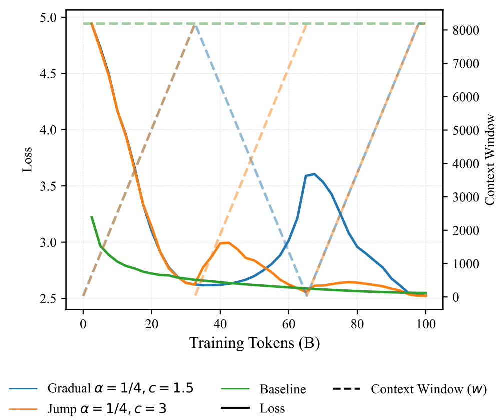


Figure 13: An illustration of the cyclic schedules with gradual increases or jumps. Dashed lines represent the context length for each step, and solid lines are the loss evaluated at an 8K length. c represents the number of cycles.

Cyclic Schedule Inspired by the cyclic schedule learning rate [45], we also wonder if cycles are helpful in the schedule. In Figure 13, we show two cyclic schedules. In the "Jump" schedule, $w(t)$ will decrease to w_s immediately after reaching L. On the other hand, the "Gradual" schedule means an "M" shape alternating between w_e and w_s. Notably, in the discontinuous Jump schedule, we notice a significant increase in long-context perplexity when we train on only short contexts for an extended period. However, as long as w increases back to L, the performance will return.

Table 19: Validation loss with different expansion rates. A box is colored red if it is significantly worse (difference > 0.01) than the best of the column. L_e is the evaluation context length. All models are of size 120M and trained on 100B tokens.

Rate ($1/\alpha$)	Tokens to Reach 8K (B)	$L_e = 512$	$L_e = 4K$	$L_e = 8K$
1	8	2.751	2.563	2.522
2	16	2.741	2.551	**2.514**
4	32	2.740	**2.551**	2.515
8	64	2.732	2.553	2.519
9	72	2.731	2.553	2.519
10	80	2.732	2.555	2.522
11	88	2.730	2.554	2.521
12	96	**2.729**	2.557	2.526
Baseline (Constant)		2.780	2.590	2.549

Table 20: Validation loss with cyclic schedules. L_e represents the evaluation context length. All models are of size 120M and trained on 100B tokens.

Type	Number of Cycles	Tokens per Cycle (B)	$L_e = 512$	$L_e = 8K$
Random			2.780	2.549
+ SkyLadder			2.732	**2.519**
Gradual	4.5	16	2.743	2.530
Jump	9	8	2.744	2.532
Gradual	2.5	32	2.732	2.521
Jump	5	16	2.733	2.521
Gradual	1.5	64	2.728	2.524
Jump	3	32	**2.727**	2.522

Table 21: Final validation loss after training 120M models on 100B tokens with different w_s when $\alpha = 1/4$ and $\alpha = 1/8$. L_e represents the context length of evaluation. A cell is colored red if its loss has a difference larger than 0.01 from the column's best. $w_s = 8192$ equals no scheduling.

w_s	$L_e = 512$	$L_e = 4K$	$L_e = 8K$
	$\alpha = 1/4$		
4	2.731	2.546	2.510
8	**2.730**	**2.545**	**2.508**
16	2.733	2.551	2.513
32	2.740	2.551	2.515
64	2.742	2.557	2.520
128	2.748	2.564	2.528
256	2.750	2.566	2.527
	$\alpha = 1/8$		
4	2.727	2.549	2.515
8	**2.725**	**2.545**	**2.510**
16	2.729	2.550	2.516
32	2.732	2.553	2.519
64	2.735	2.553	2.519
128	2.743	2.564	2.530
256	2.748	2.567	2.531
8192	2.780	2.590	2.549

In Table 20, we show that these schedules have no major impact on the final performance. This highlights that the method does not introduce additional bias in data selection: different from existing methods such as Pouransari et al. [38] that proposes to train on short data first, followed by long data, we do not assume such a curriculum on data. We argue that the context window size should be independent of the data lengths to avoid bias in training only on certain domains of data.

Initial Window Length We show the effect of having different w_s, the initial window length when the training starts. In Table 21, we show that the optimal starting length is 8 tokens. The trend is the same across both $\alpha = 1/4$ and $\alpha = 1/8$. This suggests that the starting length should be sufficiently small, irrespective of the expansion rate. It also reveals that prior studies, such as Jin et al. [21] and Pouransari et al. [38] that start with an initial length of 256 could be suboptimal.

Compute Budget We show that when the total number of tokens is limited, our method can still improve language model performance. In Table 22, we choose 12.5B, 25B, and 50B total tokens as the computing budget, and vary the expansion rate so that w reaches L at the same point during training. We observe that under different token budgets, the performance trend is the best: gradually expanding the context window gives better performance than a rapid increase.

Sliding Window Expansion A possible alternative to SkyLadder (using local causal masks by default) is to use a sliding window attention with a window size of $w(t)$ that changes with the training

Table 22: Final validation loss under different training token budgets and expansion rate α with 120M models. L_e represents the context length used for evaluation. "% of Token Budget" means how many tokens are spent in the expansion phase with $w(t)$ increasing. Under all token budgets, we observe a consistent improvement when we spend around 64% in expansion, and 36% in the stable phase.

α	Tokens to L (B)	% of Token Budget	$L_e = 512$	$L_e = 4096$	$L_e = 8192$
		Token Budget = 12.5B			
1	8	64%	**2.912**	**2.732**	**2.698**
2	4	32%	2.933	2.746	2.709
4	2	16%	2.958	2.767	2.729
8	1	8%	2.976	2.782	2.743
	Baseline		3.008	2.823	2.790
		Token Budget = 25B			
1/2	16	64%	**2.829**	**2.650**	**2.617**
1	8	32%	2.841	2.656	2.619
2	4	16%	2.851	2.665	2.626
4	2	8%	2.873	2.683	2.645
	Baseline		2.918	2.734	2.700
		Token Budget = 50B			
1/4	32	64%	**2.771**	**2.590**	**2.556**
1/2	16	32%	2.781	2.596	2.560
1	8	16%	2.789	2.603	2.564
2	4	8%	2.795	2.607	2.567
	Baseline		2.839	2.652	2.616

Table 23: Performance (%) of 1B models with different masking schemes. All models are trained on the same 100B FineWeb-Pro tokens with a final context length of 8K. Both implementations of SkyLadder outperform the baseline, and the sliding window approach excels at long tasks with a slight performance drop on standard benchmarks.

Model	Standard Avg.	Long Avg.
Random	52.5	11.1
+ SkyLadder w/ local causal	**55.2** (+2.7)	12.3 (+1.2)
+ SkyLadder w/ sliding window	54.4 (+1.9)	**12.8** (+1.7)

time. Formally, the mask becomes:

$$M_{i,j} = \begin{cases} 0 & \text{if } i - w \leq j \leq i \\ -\infty & \text{otherwise.} \end{cases}$$

so that each token in the context has a fixed preceding context of size w. When $w(t)$ reaches L, the mask becomes equivalent to a causal mask. We compare the performance of the two in Table 23 and observe that the sliding window approach shows slightly better performance in long tasks and worse performance in standard benchmarks. This is likely because overall there are more tokens with longer preceding contexts for the sliding window approach. In both cases, SkyLadder outperforms the Random baseline. We think that future work could further investigate the differences between SkyLadder implementations with causal and sliding window attention, such as the formation of attention sink [14]. There could also be possible combinations of the two: for instance, using a local mask first to disable distraction, and enabling sliding windows as the training progresses.

A.8 Additional Comparison with Related Work

We acknowledge that there are several prior works discovering a similar pattern of short-to-long pretraining. For instance, Li et al. [28] discover that using a sequence-length warmup for the initial steps in pretraining improves model stability. However, they mostly focus on stability in training loss and do not show a clear performance gain across multiple evaluations and larger scales. Moreover, we demonstrate that the benefits of scheduling a model's context window go beyond only the warmup stage. In Table 19's first row, simply warming up the model with 8B tokens results in suboptimal performance compared to a slower expansion rate. This validates that the context window should be considered as a factor to schedule over the entire training course, which also differentiates us from Li et al. [28] that only consider the warmup stage.

Another related work is Jin et al. [21] where the authors use progressive sequence lengths to accelerate training. However, their method leads to worse performance under the same token budget, while our SkyLadder shows both time saving and performance improvement with the same number of tokens. We suspect that this might be because of the suboptimal schedule they used. Moreover, their study is limited to observing the training loss of small models (up to 410M parameters), while we comprehensively show performance gain across multiple corpora, model sizes, context sizes, and a wide variety of tasks. Overall, we systematically conduct controlled experiments on the impact of context window scheduling in pretraining, providing insights to explain these previous studies.

A.9 Compute Information

We conducted all of our experiments for models with $\leq$ 1B size on an internal cluster of NVIDIA A100 nodes with 40G memory. Experiments with 3B models were conducted on H100 nodes. There are additional preliminary experiments that we did not include in the paper, which account for a fraction of the total compute. The detailed computation for each experiment is as follows: For the preliminary study on context window, pretraining a 1B model with 100B tokens (with 8K context) takes around 200 hours on a node of 8 A100s. Models of different sizes scale accordingly. For instance, plotting Figure 4(a) and (b) requires a total of 159 days of pretraining on a single node. For SkyLadder experiments, the baseline pretraining using various corpora takes the same time, and SkyLadder speeds up the training by 13% to 22% depending on the context length.

A.10 Licenses of Assets

We mainly use the following public datasets or codebases in this paper: SlimPajama [46] following the CommonCrawl Foundation Terms of Use[1], FineWeb-Pro [64] with an ODC-By 1.0 license, and TinyLlama [60] with an Apache 2.0 License.

[1] https://commoncrawl.org/terms-of-use

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract summarizes the findings of our preliminary study and the essence of the SkyLadder method: we systematically study the impact of context length on pretraining and empirically verify the effectiveness of our approach.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our study in Section A.1.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper is not a theory paper and we do not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed hyperparameter setting (Section 3.2, A.4), implementation pseudocode (A.5) and model configuration (A.3). We implement most of the experiments in TinyLlama, a popular public project for pretraining, which makes it highly reproducible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our pretraining code is based on TinyLlama, an open-source pretraining framework. The data is based on open-source pretraining corpora like SlimPajama and Fineweb-Pro. We include the code in the supplementary materials, and will open source the code upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe the training and testing setup in detail in Section 3.2 for our preliminary study, and in Section A.7.3 for SkyLadder experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We conduct McNemar test for our main results in Table 1. The details for testing are in A.7.3. Due to the excessive cost of pretraining, we do not perform repeated runs of the same setup. However, our claims are supported by multiple variations of the pretraining and comprehensive ablation studies in the main text and appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We share the compute information in Section A.9.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We fully anonymize the paper and the code. All experiments fully conform with the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impact of our work in Section A.2.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This work is not meant to release a pretrained language model or publish a dataset.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We explicitly mention and cite the source of the datasets (SlimPajama, FineWeb-Pro) and the implementation codebase in our paper (TinyLlama). License information is described in Section A.10. All of them are open-source projects available for public use.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide instructions to reproduce our experiments with the code in the supplementary materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This project does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This project does not involve human subjects or crowdsourcing.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We did not use LLMs for developing our method.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.