Data Engineering for Scaling Language Models to 128K Context

Yao Fu [1] **Rameswar Panda** [2] **Xinyao Niu** [3] **Xiang Yue** [4] **Hannaneh Hajishirzi** [5] **Yoon Kim** [6] **Hao Peng** [7]

Abstract

We study the continual pretraining recipe for scaling language models' context lengths to 128K, with a focus on data engineering. We hypothesize that long context modeling, in particular *the ability to utilize information at arbitrary input locations*, is a capability that is mostly already acquired through large-scale pretraining, and that this capability can be readily extended to contexts substantially longer than seen during training (e.g., 4K to 128K) through lightweight continual pretraining on appropriate data mixture. We investigate the *quantity* and *quality* of the data for continual pretraining: (1) for quantity, we show that 500 million to 5 billion tokens are enough to enable the model to retrieve information anywhere within the 128K context; (2) for quality, our results equally emphasize *domain balance* and *length upsampling*. Concretely, we find that naïvely upsampling longer data on certain domains like books, a common practice of existing work, gives suboptimal performance, and that a balanced domain mixture is important. We demonstrate that continual pretraining of the full model on 1B-5B tokens of such data is an effective and affordable strategy for scaling the context length of language models to 128K. Our recipe outperforms strong open-source long-context models and closes the gap to frontier models like GPT-4 128K.

1. Introduction

A context window of 128K tokens enables large language models to perform tasks that significantly beyond existing paradigm, such as multi-document question answering (Caciularu et al., 2023), repository-level code understanding (Bairi et al., 2023), long-history dialog modeling (Mazumder & Liu, 2024), and language model-powered autonomous agents (Weng, 2023). A popular testbed for whether models can actually utilize long context length is the recent Needle-in-a-Haystack test (Kamradt, 2023), which asks the model to precisely recite the information in a given sentence where the sentence (the "needle") is placed in an arbitrary location of a 128K long document (the "haystack"). In the open-source space, although works like LongLoRA (Chen et al., 2023b) and YaRN-Mistral (Peng et al., 2023) theoretically support 100K context, they are not able to pass this test at such context lengths, as shown in Fig. 1. Currently, only closed-source frontier models like GPT-4 128K have demonstrated strong performance on the Needle-in-a-Haystack test.

This work investigates data engineering methods for scaling language models' context lengths. Our objective is to continue pretraining the language model on appropriate data mixtures such that it can pass the Needle-in-a-Haystack test at 128K length. Given that most existing models are trained on less than 4K context length (Touvron et al., 2023a) and that attention has quadratic complexity, continual pretraining with full attention on much longer context lengths (we train on 64K-80K context lengths) may seem prohibitively costly at a first glance. However, we show that this is feasible under academic-level resources (see Table 2). We use LLaMA-2 7B and 13B as our base models. We do not make any significant change to model architecture other than adjusting the base of RoPE, as in Xiong et al. (2023). Our major focus is the data recipe: *what* and *how much* data is able to well-adapt a model to pass the Needle-in-a-Haystack test at 128K context length.

We hypothesize that the capability to utilize information at arbitrary locations within long context length is (mostly) already acquired during pretraining, even for models pretrained on substantially shorter 4K contexts. This hypothesis is in contrast to existing works like Xiong et al. (2023); XVerse (2024), which perform continual pretraining on a large amount of data (400B tokens) to *inject* long-context-modeling capabilities; in this strategy, the cost can be as high as pre-training from scratch. In this work we show that continual pretraining on a small amount of long-context data, in our case, 1-5B tokens, can "unlock" a 7B model's

[1]University Edinburgh [2]MIT-IBM Watson AI Lab [3]University of Melbourne [4]CMU [5]University of Washington [6]Massachusetts Institute of Technology [7]University of Illinois at Urbana-Champaign. Correspondence to: Yao Fu <yao.fu@ed.ac.uk>, Yoon Kim <yoonkim@mit.edu>, Hao Peng <haopeng@illinois.edu>.

Needle-in-a-Haystack sentence retrieval test, comparison between our method v.s. baselines v.s. GPT-4
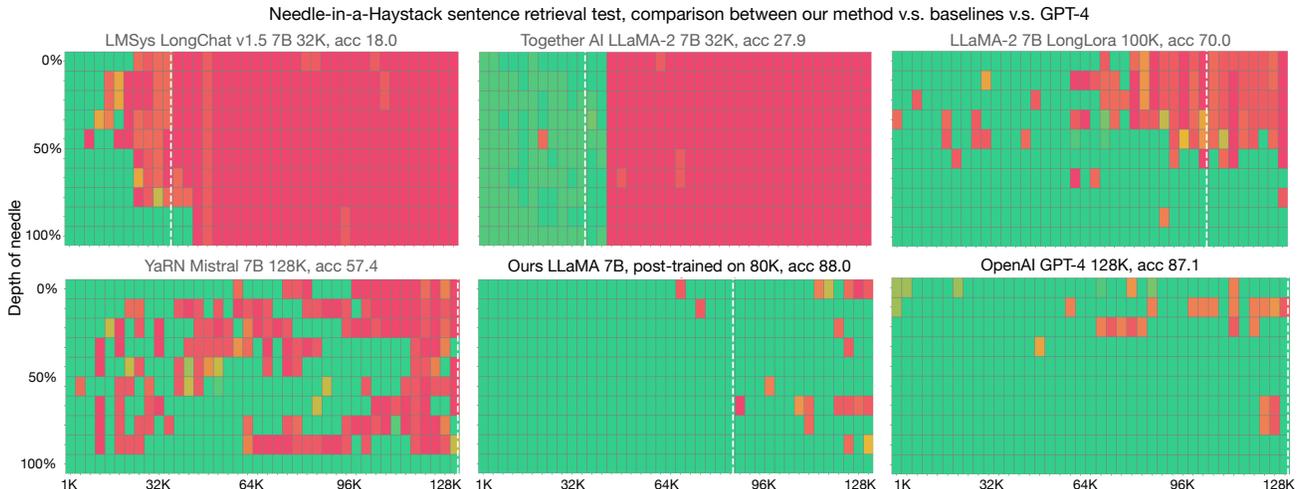
Figure 1. Needle-in-a-Haystack test (Kamradt, 2023) performance comparison. The x-axis denotes the length of the document (the "haystack"); the y-axis indicates the position that the "needle" (a short sentence) is located within the document, from 1K to 128K tokens. For example, 50% indicates that the needle is placed in the middle of the document. A red cell means the the language model cannot recite the information in the needle, and a green cell means the model can. A white dashed line means models' continual pretraining (or finetuning for instruction-tuned models) context length; thus the area to its right indicates length generalization. Most existing open-source models make mistakes when the document is long. Our post-training recipe demonstrates strong performance up to about 100K length.

capability of precise retrieval over much longer context lengths than seen in original pretraining.

We further show that upsampling long sequences while retaining the domain mixture of the pretraining corpora is crucial for context scaling, yet overlooked by existing works (Table 1). Most existing works are based on the following intuition: to model long-range dependencies one needs long sequence data, which domains like books provide; therefore, it is necessary to upsample domains containing long sequences in the data mixture, as done by LongChat 32K (Li et al., 2023a) and YaRN Mistral 128K (Peng et al., 2023). However, we show that this intuitive solution is suboptimal because, as we observe, this results in perplexiy degradations in other domains (Table 5). Instead, a data mixture that keeps the domain mixture ratio the same as the pretraining mixture, and then upsampling long sequences within each domain gives the most stable performance gain. We give evidence that this is the primary reason our solution improves long context tasks while maintaining short context performance, compared with strong baselines like YaRN-Mistral 128K (Peng et al., 2023) and LongLoRA 100K (Chen et al., 2023b).

In summary, we propose a concrete data recipe for scaling language model context length to 128K, specifically, which involves continual pretrain the full-attention model on 1-5B tokens of per-source-length upsampled data. We show that our recipe results in 7B and 13B LLaMA-2 of strong long-context performance, substantially closing the gap to frontier models like GPT-4 128K on the Needle-in-a-Haystack test, opening future possibilities of studying

long-context modeling under academic budgets.

2. Background

Frontier language models feature extremely long context lengths, such as OpenAI's GPT-4 Turbo 128K (Nov 2023) and Anthropic's Claude-100K (May 2023). In the regime of 100K, a wide array of new applications emerge, such as repo-level code understanding (Bairi et al., 2023), long-history dialog modeling (Mazumder & Liu, 2024), and language model-powered autonomous agents (Weng, 2023). A recent testbed for long-range capabilities is the Needle-in-a-Haystack benchmark, first proposed by Kamradt (2023) in Nov 2023. This benchmark asks the language model to recite the information in a "needle" sentence ("The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day") that is randomly inserted at an arbitrary location in a long essay. Since its release, it has become a popular sandbox for testing whether models can utilize 100K+ context lengths, as it differentiates models that can precisely recite the given information at arbitrary input location versus those models that cannot. So far there is neither public knowledge nor open-source work about achieving precise retrieval anywhere at this scale to the best of our knowledge.

This work improves the long-context capability over strong open-source baselines, and closes the gap to GPT-4 on the Needle-in-a-Haystack benchmark, as demonstrated in Fig. 1. Our baselines here include LMSys' LongChat v1.5 32K (Li et al., 2023a), Together Compute's LLaMA-2 32K (Together, 2023), YaRN Mistral 128K (Peng et al., 2023), and

Table 1. Major differences between our method v.s. existing work from a data perspective, which we view as critical for extending context length. Our data mixture differs from previous work in three ways: (1) *length of continual pretraining data*: we use 80K compared to Together's 32K, which does not generalizes beyond 32K; (2) *data mixture*: we use SlimPajama which has balanced domains compared to LongChat, which uses dialog-only; (3) *length upsampling*: we upsample long sequences compared to LongLoRA, which does not. Despite the subtlety of these details (e.g., many of these details are just mentioned as a single line in previous works), we find that these details are crucial for performance on long-range retrieval.

	Existing data solution	**Our solution**
Together LLaMA-2 32K	Train on 32K length	Train on 80K length
LongChat v1.5 32K	Train on 32K length dialog data	Train on 80K length-upsampled SlimPajama
YaRN Mistral 7B 128K	Train on 16K without length-upsampling	Trained on length-upsampled SlimPajama
LongLoRA 100K	Train on Redpajama no length upsampling	With length upsampling

LongLoRA (Chen et al., 2023b), which are so far the top open-source long-context language models. These works focus on different aspects of long-context language modeling. For example, YaRN (Peng et al., 2023) focuses on positional embeddings, LongLoRA (Chen et al., 2023b) focuses on efficient attention, and Together (Together, 2023) focuses on a full-stack solution. Our work focuses on data-engineering, and identifies critical data aspects for extending language models' long-term dependency modeling to much longer contexts than seen during regular pretraining.

The major differences between this work and existing work are listed on Table 1. Together's LLaMA-2 is trained on 32K, but only generalizes to about 40K length. LongChat is trained on dialog-only data; later in Table 5 we will show that improvements on one domain has limited transfer to other domains, indicating that one should consider a balanced mixture of different domains. LongLoRA does not upsample long sequences; later in Fig. 4 we will show that upsampling long sequences is critical for precise retrieval. These details are relatively hard to notice, yet we believe that they are the important for improved performance. Before the Needle-in-a-Haystack test, most existing works use test negative log-likelihood as evaluation, effectively concealing the underlying differences beyond low loss. We show that, similar test loss could result in substantially different behavior when performing precise retrieval (Fig. 4).

Another important related work is the previous LLaMA Long (Xiong et al., 2023) work and the concurrent XVERSE (XVerse, 2024) work, which continue pretraining the model on 32K sequences for about 500 billion tokens. These works are implicitly motivated by the view that long-context modeling is a new capability that must be "injected" through large-scale training. We instead hypothesize that the base model has mostly already acquired this capability through large-scale pretraining, and thus a lightweight continual pretraining on relatively small data (e.g., 5B tokens) is enough to extend these capabilities to much longer context lengths (Fig. 3).

3. Long Context Data Composition

We use the SlimPajama (Soboleva et al., 2023) dataset for continual pretraining. This dataset is an open-source reproduction of the LLaMA (Touvron et al., 2023a) pretraining data mixture, consisting of 82% web data (67% from CommonCrawl and 15% from C4), 4.5% code (Github), 4.5% Wikipedia, 4.5% books, 2.5% Arxiv, and 2.0% Stack-Exchange. Since this dataset closely mirrors that used to pretrain the LLaMA models, there is less concern of distribution shift during continual pretraining; it is therefore used by many recent works like Fuzhao Xue & You (2023).

The documents' lengths and their source domains are two closely related confounding factors in data engineering because long data usually come from particular sources. Our examination shows books and Github code are the longest sources, followed by Arxiv. Webpages like C4 and Stack-Exchange tend to be shorter. Directly upsampling long data changes the domain mixture, e.g., upsampling sequences longer than 100K will increase the portion of the books domain. Likewise, changes in the domain mixture will result in shifts of the length distribution. Our guideline is to decompose these two factors step by step, as is shown in Fig. 2. We consider the following methods:

Cut at 4K: This truncates documents into 4K-length chunks. This approach inherits the original data mixture and is used by most pretraining works, such as (Touvron et al., 2023a; Hoffmann et al., 2022). Since there are about 30% documents that are naturally longer than 4K, this approach breaks such naturally-existing long-range dependencies.

Cut at 128K: This preserves most of the naturally-existing long-range dependencies without changing the domain mixture. LongLoRA (Chen et al., 2023b) follows this approach. However, we will show later that only using the naturally-existing long dependencies is inadequate (Fig. 4).

Per-source Upsampling: This retains the domain mixture, then upsamples long documents within each domain. This approach upsamples long documents without chang-
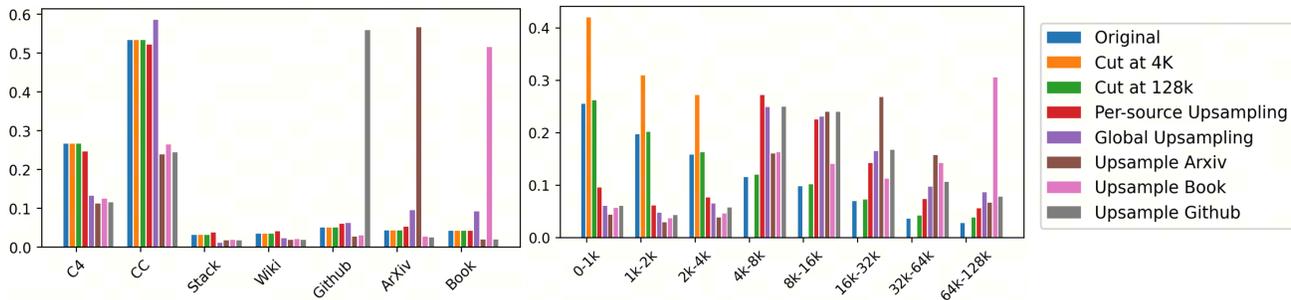
Figure 2. Length and domain distributions of the various data mixture strategies. We use SlimPajama (Soboleva et al., 2023), an open-source reproduction of LLaMA (Touvron et al., 2023a) pretraining data mixture, as the source dataset. The original data mixture has about 30% of documents that are naturally longer than 4K. Cutting documents at 4K, the common practice of pretraining like Touvron et al. (2023a), breaks such long-range dependencies. Cutting documents at 128K retains the naturally-existing long-range dependencies. Global upsampling longer sequences slightly changes the data mixture. Per-source upsampling longer sequences increases the portion of long sequences while keeping the domain mixture the same. Upsampling Arxiv / Book / Github data simultaneously changes domain and length distributions.

ing the domain mixture. We recommend this approach as our experiments suggest that this gives the most balanced performance gain.

Global Upsampling: This upsamples long documents while ignoring their source domains, and consequently slightly changes the domain mixture. Together (2023) uses this approach.

Upsample Arxiv/ Book/ Github: This intentionally upsamples Arxiv/Book/Github data, assuming that these domains are more likely to contain naturally-occurring long documents. For example, MPT-storyteller (Team, 2023) upsamples books. This approach changes both the domain and length distributions.

The above approaches generally cover most data recipes used in prior works discussed in section 2. In our experiments, we will show that: (1) using the original mixture cutting at 128K is insufficient for precise retrieval over long-context (Fig. 4), and one need to upsample long sequences; (2) improved performance in one domain may not transfer and could even hurt another domain (Table 5), showing that one needs to balance the mixture ratio of different domains.

4. Infrastructure and Engineering

We continue pretraining the model on the data mixture yielded by the "Per-source Upsampling" strategy as introduced in section 3, where we train with a 80K sequence length for 7B models, and 64K for 13B models. A "brute force" training on 80K context may seem impractical at first glance given the quadratic complexity of attention. However, in our initial tests, we found this to be feasible, and the actual wallclock time was far from quadratic. This is due to the fact that most of the time is spent on data transfer from CPU to GPU (since we use offloading), from one GPU to another GPU via NVLink (since we use Zero3, see Rajbhan-

dari et al., 2020) and from GPU High-Bandwidth Memory (HBM) to Streaming Multiprocessors (SM) (as in FlashAttention, see Dao, 2023). These IO operations are all constant or linear in sequence length. The actual quadratic attention computation, when performed on SM, is highly parallelized, and thus largely hidden by the linear IO cost. Consequently, training on 80K is only 3x slower than training on 4K.

Our specific configuration is listed on Table 2. We note that this configuration is substantially cheaper than previous work (Xiong et al., 2023), as they continue pretraining the model on more than 400B tokens. However this configuration seems to be the limit of Huggingface-DeepSpeed framework, and setting even longer context leads to memory overflow. Since Zero Optimizer is data parallel, increasing number of GPUs cannot further increase context length, and most of the GPU memory is taken by the KV cache. We refer the reader to more advanced sequence parallelism techniques in (Li et al., 2021; Jacobs et al., 2023) for training on even longer sequences (e.g., 200K), and we leave the exploration of training on 200K context for future work.

For training, we use a constant learning rate 2e-5. We modify the base of RoPE positional encoding to adjust it to longer context, as in Xiong et al. (2023). We pack all data to 80K chunks regardless of the document boundary, following common practice (Raffel et al., 2020; Touvron et al., 2023a). We set the batch size to be 4M tokens. Note that this batch size is the same as training on 4K context length, as we increase the length of a chunk but decrease the number of chunks in a batch. We train the model on 5B tokens, which translates to 5B (size of data) / 4M (batch size) = 2000 optimization steps.

5. Experimental Results

We continue pretraining the base LLaMA 2 7B/ 13B models on data produced by our "Per-source Upsampling" strategy

Table 2. We show that long context continual pretraining is feasible under academic-level resources. Our configuration on 8×80G A100s, as listed below, takes 5 days, which is about 1% budget than existing works such as Xiong et al. (2023), which trains on 400B tokens.

Framework	Huggingface Transformers
	+ DeepSpeed Zero 3
	+ FlashAttention 2
	+ Gradient Checkpointing
	+ CPU Offloading
Hardware	8 × 80G A100
LLaMA 2 7B	Ctx. 4K 3 days / 10B tokens
	Ctx. 80K 10 days / 10B tokens
LLaMA 2 13B	Ctx. 4K 5 days / 10B tokens
	Ctx. 64K 13 days / 10B tokens
Hardware	2 × 8 × 80G A100
LLaMA 2 7B	Ctx. 4K 2 days / 10B tokens
	Ctx. 80K 7 days / 10B tokens
LLaMA 2 13B	Ctx. 4K 4 days / 10B tokens
	Ctx. 64K 10 days / 10B tokens

Table 3. Our method achieves better long context task performance (Needle. performance) *without* compromising short context performance (MMLU).

	Ctx.	Needle.	MMLU
Non-LLaMA Models			
GPT-4-Turbo	128K	**87.1**	**86.4**
GPT-3.5-Turbo	16K	-	67.3
YaRN Mistral 7B	128K	57.4	59.4
LLaMA-2 7B Based Models			
Together LLaMA-2 7B	32K	27.9	**44.8**
LongChat v1.5 7B	32K	18.0	42.3
LongLoRA 7B	100K	70.0	37.9
Ours LLaMA-2 7B	80K	**88.0**	43.3
LLaMA-2 13B Based Models			
LongLoRA 13B	64K	54.1	50.1
Ours LLaMA-2 13B	64K	**90.0**	**52.4**

lengths are mostly around 10K, which were considered long at their release time, but substantially shorter than the 128K regime, which is the focus of the present work.

5.1. Overall Performance

Figure 1 compares our method with Together AI's LLaMA-2 32K (Together, 2023), LMSys' LongChat v1.5 32K (Li et al., 2023a), YaRN Mistral 7B 128K (Peng et al., 2023), and LongLoRA 100K (Chen et al., 2023b). We choose these models as our baseline because they are so far the top open-source long-context language models (as evidenced by thousands of GitHub stars). LongChat v1.5 32K does not perform well even for sequence length 16K-32K, and we hypothesize that this is because the model has not gone through enough continual pretraining, but directly goes to finetuning. Together AI's LLaMA-2 7B's data mixture is similar to ours, which we believe to be the reason that they perform well within 32K. Yet their model does not generalize beyond 32K, while ours generalizes from 80K to 128K. For LongLoRA, we believe the two major problems are that they use sparse attention and that they do not upsample long sequence data, which we show is problematic in Fig. 4. For YaRN Mistral, we hypothesize that its underperformance is due to its being only trained on PG19 book (Rae et al., 2019); we show that it is important to use a balanced domain mixture in Table 5.

In Table 3 we show that our method not only improves precise retrieval, but maintains short context performance, evidenced by strong MMLU (Hendrycks et al., 2020) score, which is a widely-accepted benchmark for testing the general capabilities (within short context) of language models, and is used by Chinchilla (Hoffmann et al., 2022), LLama (Touvron et al., 2023a;b), Mistral (Jiang et al., 2023),

(Sec. 5.3). The number of training tokens is 5B. We first show that our method outperforms strong open-source baselines like YaRN Mistral 7B 128K (Peng et al., 2023) and LongLoRA 7B 100K (Chen et al., 2023b), and closes the gap to GPT-4 Turbo 128K. Then we dissect the ingredients of our data recipe. In section 5.2, we discuss data sufficiency (the 5B part) and show that as we increase the amount of data from 100M to 1B, the model gradually surfaces the ability to precisely retrieve given information at arbitrary locations within 128K documents. In section 5.3, we discuss the data mixture (the per-source length upsampled part), and show that the original mixture of SlimPajama is insufficient (although it has about 30% data longer than 4K), and one needs to upsample long sequences for the model to acquire the long context capability. We also show that the improvements on one upsampled domain (e.g., Books), have limited transfer to others, and can even hurt them. The Book and Code domains are a particular example that improvements on one come with detrimental impact on the other.

Existing work heavily uses validation loss as the primary evaluation (Chen et al., 2023a;b; Xiao et al., 2023; Peng et al., 2023). We show that using only the validation loss is insufficient because two data recipes that result in similar loss may have substantially different retrieval capabilities, as we show in Fig. 4. In addition to Needle-in-a-Haystack, we use a real-world book-long question answering benchmark (Zhang et al., 2023) for our evaluation. There are earlier long context benchmarks such as Zeroscrolls (Shaham et al., 2023), longbench (Bai et al., 2023b) and L-Eval (An et al., 2023). We do not evaluate on them because their

Figure 3. As we increase the number of trained tokens, the model's retrieval performance increases with converged validation loss. We further note two observations with regard to data quantity: (1) 500M tokens are enough to unlock most of the retrieval accuracy. Since we are not explicitly training the model to perform retrieval, this potentially suggests that the model already has the precise retrieval capability, and that we are simply extending this capability with lightweight continual pretraining; (2) the model's retrieval performance saturates at about 5B tokens, and further scaling to 10B tokens does not improve length generalization.

QWen (Bai et al., 2023a), and Baichuan (Yang et al., 2023a).

Table 4 further compares 128K context language models on a book-long question answering benchmark recently proposed by Zhang et al. (2023). This task conditions language models on a book and then asks questions about the plot. Our method outperforms LongLoRA and Yarn Mistral (even though Mistral 7B is a stronger base model than LLaMA 2 7B we use). Our 13B model performance closes the gap to GPT-4 128K, and we anticipate that future scaling and instruction tuning will further improve performance. While there are other long-context benchmarks in InfiniBench (Zhang et al., 2023), in our initial experiments we found that models often had trouble understanding the instruction (because they are not instruction tuned). Hence we focus on the BookQA benchmark where base LLMs performed reasonably without instruction tuning.

5.2. Data Quantity

Now we consider data quantity, i.e., how much data enables the language model to utilize the information at arbitrary locations within 128K context. Our hypothesis is that precise retreival over long-range context is an intrinsic capability obtained by large-scale pretraining, even when the pretraining context length is substantially shorter (4K in many cases). If this hypothesis is true, then lightweight continual pretraining should be enough to extend this capability to much longer context lengths than see in training. That is, we would not need data-intensive continual pretraining as used by Xiong et al. (2023) and XVerse (2024).

Table 4. The performaince gain shown in Needle-in-a-Haystack further translates to improvements on downstream BookQA. We report the results on 128K length InfiniBench long book question answering task (Zhang et al., 2023). Our method outperforms top open-source models and closes the gap to GPT-4.

Model / Train Len	Test Len	BookQA
GPT-4-Turbo 128K	128K	**37.4**
LongLoRA 7B 100K	128K	24.3
Ours LLaMA-2 7B 80K	128K	27.4
LongLoRA 13B 64K	128K	24.6
YaRN Mistral 7B 128K	128K	26.3
Ours LLaMA-2 13B 64K	128K	**31.1**

Fig. 3 shows how data scaling gradually unlocks the model's retrieval capability. When the model is continually pretrained on 100M to 300M tokens, the model's loss gradually decreases, but it has not converged yet. At 500M to 1B tokens, the model achieves relatively good performance within its continually pretrained 80K context, but does not generalize to 80K-128K range. After 5B tokens, the model performs well on 0-80K, and can generalize to unseen lengths 80K-128K. At 10B token, the model seems to overfit on its 80K training range, and the length generalization starts to decrease. Along this process, we see a gradual decrease in loss, which correlates with the increase in the retrieval performance. This result provides more evidence for our hypothesis, as only 500M tokens are required to enable a model's retrieval capability with 80K context.

Continually pretraining a 7B LLaMA-2 takes about 5 days on 8×80G A100 GPUs, and the cost is reasonable compared

Table 5. How different data engineering methods improves (or does not improve) the original data mixture, where we show results from continual pretraining of a 7B LLaMA-2 on 5B tokens packed to 80K length. We consider how changing the original data mixture to long-context upsampled data mixture results in different loss across domains, and report the *loss differences* to the original data mixture (e.g., the loss of per-source length upsampling minus the loss of original mixture). We view a loss difference larger than 0.01 significant, marked by a red shade indicating performance decrease of larger than +0.01 loss; or a green shade indicating performance improvements of smaller than -0.01 loss, or a grey shade indicating no significant differences. Although upsampling book / code / arxiv improves in-domain performance for both short and long context, such improvements do not transfer to all domains. Instead, upsampling one domain, e.g., code, may even harm another domain, e.g., book. *Per-source length upsampling is the most balanced mixture* with almost no significant increase of loss across domains.

	C4	CC	Stack	Arxiv	Wiki	Book	Github
0 - 4K Context Length							
Original	2.038	1.760	1.519	1.660	1.424	2.085	0.907
v.s. Per-source	+ .002	+ .008	− .001	− .008	− .040	− .065	− .008
v.s. Global	+ .008	+ .010	+ .015	− .020	− .020	− .140	+ .015
v.s. Code↑	+ .010	+ .016	+ .010	+ .006	− .026	+ .030	− .023
v.s. Book↑	+ .010	+ .016	+ .021	+ .000	− .010	− .175	+ .029
v.s. Arxiv↑	+ .006	+ .016	+ .013	− .060	− .030	+ .040	+ .025
4K - 128K Context Length							
Original	1.560	1.650	0.786	1.075	1.313	1.852	0.447
v.s. Per-source	− .010	− .010	− .006	− .011	− .044	− .014	+ .002
v.s. Global	− .010	− .006	− .001	− .016	− .040	− .018	− .007
v.s. Code↑	− .008	− .002	− .003	− .007	− .042	− .010	− .029
v.s. Book↑	− .010	− .006	+ .001	− .007	− .037	− 0.30	+ .000
v.s. Arxiv↑	− .008	− .002	+ .002	− .036	− .039	− .010	− .004

to large-scale pretraining (which requires months of compute on thousands of GPUs). Our results suggest that for supervised finetuning, since training on long-context is substantially cheaper than previously thought, future work may dive deeper on the solutions for 100K length finetuning and reasoning, which so far has almost no open-source work to our knowledge. For pretraining research, currently there is no definite answer as to whether long-context continual pretraining should be combined with other capabilities, such as math (Azerbayev et al., 2023) and code (Chen et al., 2021), which typically require hundreds of billions of tokens. Our results suggest that long-context continual pretraining could be a separate stage after code and math pretraining.

5.3. Data Mixture

Now we discuss why *per-source length upsampling* is necessary when constructing the data mixture. Recall that this strategy keeps the mixture ratio of the data sources the same as the original data, i.e., 67% CommonCrawl (CC), 15% C4, 4.5% Github, 4.5% Wikipedia, 4.5% books, 2.5% Arxiv and 2.0% StackExchange for SlimPajama. Then in each of the domains, we upsample sequences longer than 4K from about 30% to about 70%. Doing this enables us to keep the domain mixture ratio fixed, only changing the length

distribution of the training documents. In contrast, globally upsampling long sequences (without considering their domain), or intentionally upsampling code/ book/ Arxiv (since they are long) changes both the domain mixture and the length distribution.

We compare all the data mixtures with the original one packed to 80K / 64K for the continual pretraining of the 7B / 13B LLaMA-2. This baseline approach effectively never changes the original data, but keeps the naturally existing long sequence that would otherwise been cut to 4K (recall Sec. 3). If a data mixture is effective, it should be at least better than this baseline "do nothing" approach.

Table 5 compares the per-domain loss differences of all the data mixture against the baseline original mixture. We report the differences of the validation loss, where a more than 0.01 loss change is considered significant, following common pretraining practice (Kaplan et al., 2020; Peng et al., 2023; Hoffmann et al., 2022). We observe: (1) for 0-4K short context data, most of the data mixtures have a negative impact on webpages (C4, CC and StackExchange), except for the per-source approach; (2) performance improvements from domains like Book may not transfer to others, and even hurt code performance; (3) per-source length upsampling is the most balanced mixture that improves 4K-128K
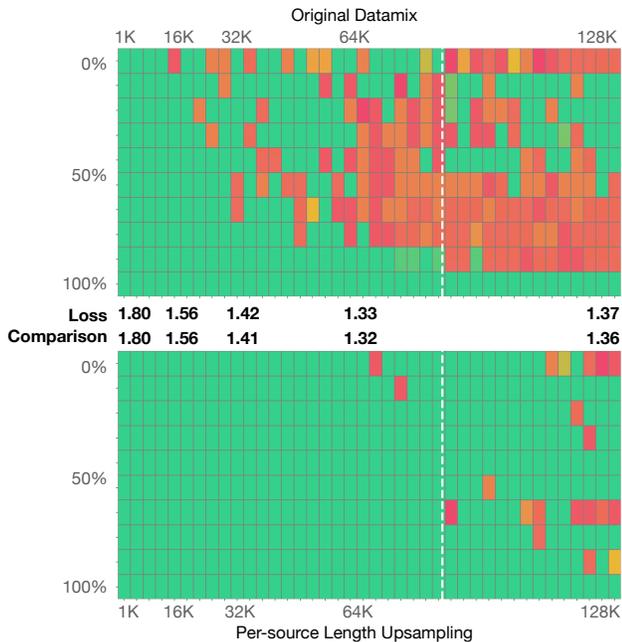
Figure 4. The importance of length upsampling. The validation loss, a common measure for long-context language modeling used in Xiong et al. (2023); Chen et al. (2023b), cannot reveal long-context retrieval capability, but Needle-in-a-Haystack can. The original data mixture, i.e. without length upsampling, despite giving very close loss, performs badly on precise retrieval. Per-source length upsampling significantly improves precise retrieval.

context losses without much sacrificing short-context losses, whereas all other methods show more or less performance tradeoff across domains. Combining these observations, we recommend the per-source length-upsampling strategy.

Since per-source length upsampling keeps the domain mixture ratio while increasing tokens from long sequences, its major difference to the original data mixture is the distribution of sequence length. Fig. 4 makes a close comparison between the two data strategies. Specifically, we train two LLaMA-2 7B models with a 80K context length using original / per-source upsampled data mixture, and report their per-length validation loss and Needle-in-a-Haystack performance. Note that LongLoRA (Chen et al., 2023b) uses the original data mixture without length upsampling, so our results also explains why we achieve better performance than LongLoRA (Fig. 1). We see that the original data mixture without length upsampling, despite achieving a very close loss, underperforms on precise retrieval. Per-source length upsampling significantly improves precise retrieval. This observation also serves as strong evidence why only using test loss, the evaluation used in most prior work (Chen et al., 2023a; Peng et al., 2023; Chen et al., 2023b; Xiao et al., 2023; Anthropic, 2023), may conceal the underlying model differences. The necessity of length upsampling is demonstrated by the Needle-in-a-Haystack test.

6. Discussion

We attribute our improvements over strong open-source baselines, as is detailed in section 5, to our careful treatments of data engineering. Our results echo the recent wisdom that in large language model research, data engineering is equally important as modeling and algorithmic innovations (Kaplan et al., 2020; Hoffmann et al., 2022; Brown et al., 2020). As we list in Table 1, many of the data details are crucial for strong long-context performance, yet may be easily overlooked. We also acknowledge that our research is made possible through utilizing the latest innovations in machine learning systems research, particularly FlashAttention (Dao, 2023) as it reduces the memory usage of the attention mechanism from quadratic to linear. Incorporating further work on sequence parallelism (Li et al., 2021; Jacobs et al., 2023) could enable brute force training of even longer context (e.g., 200K) models.

The Transformer's use of position embeddings makes it difficult to generalize significantly beyond contexts seen during training even with relative positional encodings (Press et al., 2021; Sun et al., 2022; Li et al., 2023b), thus necessitating (at least a little bit of) continual pretraining. There has much recent work on RNN-like architectures, which implicitly encode positional information through their hidden states, that perform competitively with Transformers (Sun et al., 2023; Qin et al., 2023; Gu & Dao, 2023; Yang et al., 2023b). It would be interesting to test whether such models can generalize zero-shot to longer contexts than seen during training on the Needle-in-a-Haystack benchmark.

Long-context language model research at the 100K-level is still a developing research area. This work only studies continual pretraining, and research on instruction finetuning language models on tasks of 100K context length (e.g., repo-level code understanding) is still limited. So far there seems to no open-source instruction-finetuned 100K context language models. We hope our work serve as a basis for future work on 100K-level long context superivsed finetuning.

7. Conclusion

This work studies a continual pretraining recipe for scaling language models' context length to 128K tokens. We demonstrate that the ability to utilize information at arbitrary locations within the 128K input is already mostly acquired by large-scale pretraining, even for models pretrained on substantially shorter 4K context. We show that continual pretraining of the full model on 1-5 billion tokens from per-source length-upsampled data gives the most balanced performance improvements. Our work closes the gap to frontier models like GPT-4 128K on the retrieval task and serves as the foundation for future long-context instruction finetuning research.

Impact Statement

This work studies data-engineering for extending language models to 128K context length. There are multiple applications of such long context, for example, multi-document question answering, repository level code understanding, long-history dialog modeling, and language model based autonomous agents. Many frontier models, such as GPT-4, Claude-2, Character.AI have already deployed commercial closed source long context language models. We believe that open-sourcing such technologies will improve community's understanding of large long context language models and further advance AI research. There might also be certain safety concerns (as the the case for all large language models) and the model needs to be aligned to reject unsafe queries.

References

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

Anthropic. Model card and evaluations for claude models, July 2023. URL https://www.anthropic.com/product.

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023a.

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

Bairi, R., Sonwane, A., Kanade, A., Iyer, A., Parthasarathy, S., Rajamani, S., Ashok, B., Shet, S., et al. Codeplan: Repository-level coding using llms and planning. *arXiv preprint arXiv:2309.12499*, 2023.

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

Caciularu, A., Peters, M. E., Goldberger, J., Dagan, I., and Cohan, A. Peek across: Improving multi-document modeling via cross-document question-answering. *arXiv preprint arXiv:2305.15387*, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

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

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Fuzhao Xue, Zian Zheng, Y. F. J. N. Z. Z. W. Z. and You, Y. Openmoe: Open mixture-of-experts language models. https://github.com/XueFuzhao/OpenMoE, 2023.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. 2023. URL https://api.semanticscholar.org/CorpusID:265551773.

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

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jacobs, S. A., Tanaka, M., Zhang, C., Zhang, M., Song, L., Rajbhandari, S., and He, Y. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

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

Kamradt, G. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Li, D., Shao, R., Xie, A., Sheng, Y., Zheng, L., Gonzalez, J. E., Stoica, I., Ma, X., and Zhang, H. How long can open-source llms truly promise on context length?, June 2023a. URL https://lmsys.org/blog/2023-06-29-longchat.

Li, S., Xue, F., Baranwal, C., Li, Y., and You, Y. Sequence parallelism: Long sequence training from system perspective. *arXiv preprint arXiv:2105.13120*, 2021.

Li, S., You, C., Guruganesh, G., Ainslie, J., Ontanon, S., Zaheer, M., Sanghai, S., Yang, Y., Kumar, S., and Bhojanapalli, S. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*, 2023b.

Mazumder, S. and Liu, B. *Lifelong and Continual Learning Dialogue Systems*. Springer Nature, 2024.

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

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

Qin, Z., Yang, S., and Zhong, Y. Hierarchically gated recurrent neural network for sequence modeling. *CoRR*, abs/2311.04823, 2023. doi: 10.48550/ARXIV.2311.04823. URL https://doi.org/10.48550/arXiv.2311.04823.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

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

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Shaham, U., Ivgi, M., Efrat, A., Berant, J., and Levy, O. Zeroscrolls: A zero-shot benchmark for long text understanding, 2023.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J. R., Hestness, J., and Dey, N. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X., and Wei, F. A length-extrapolatable transformer. *arXiv preprint arXiv:2212.10554*, 2022.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Team, M. N. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL www.mosaicml.com/blog/mpt-7b. Accessed: 2023-03-28.

Together. Llama-2-7b-32k-instruct — and fine-tuning for llama-2 models with together api, August 2023. URL https://www.together.ai/blog/llama-2-7b-32k-instruct.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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

Weng, L. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023. URL https://lilianweng.github.io/posts/2023-06-23-agent/.

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

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

XVerse. Xverse-13b, January 2024. URL https://github.com/xverse-ai/XVERSE-13B/blob/main/README_EN.md.

Yang, A., Xiao, B., Wang, B., Zhang, B., Bian, C., Yin, C., Lv, C., Pan, D., Wang, D., Yan, D., et al. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*, 2023a.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023b.

Zhang, X., Chen, Y., Hu, S., Wu, Q., Chen, J., Xu, Z., Dai, Z., Han, X., Wang, S., Liu, Z., and Sun, M. Infinitebench: 128k long-context benchmark for language models, 2023.