LongAlign: A Recipe for Long Context Alignment of Large Language Models

Anonymous ACL submission

Abstract

Extending large language models to effectively handle long contexts requires instruction fine-tuning on input sequences of similar length. To address this, we present LongAlign—a recipe of the instruction data, training, and evaluation for long context alignment. First, we construct a long instruction-following dataset using Self-Instruct. To ensure the data diversity, it covers a broad range of tasks from various long context sources. Second, we investigate different strategies to speed up supervised fine-tuning on datasets with uneven length distribution, namely packing and sorted batching. Additionally, we develop a loss weighting method to balance the contribution to the loss across different sequences during packing training. Third, we introduce the LongBench-Chat benchmark for evaluating instruction-following capabilities on queries of 10k-100k in length. Experiments show that LongAlign outperforms existing recipes for LLMs in long context tasks by up to 30%, while also maintaining their proficiency in handling short, generic tasks.

1 Introduction

Large language models (LLMs) with large context windows facilitate tasks such as summarization, question answering on long text and code (Bai et al., 2023a). Importantly, they may form the foundational support for life-long conversations and complex agent scenarios (Xiao et al., 2023; Liu et al., 2023). Existing works to build long-context LLMs predominantly focus on context extension (Chen et al., 2023a; Xiong et al., 2023; Peng et al., 2023), that is, position encoding extension and continual training on long text.

In this work, we instead focus on the perspective of long context alignment, i.e., instruction fine-tuning LLMs to handle long user prompts. However, several challenges are required to address. First, there is an absence of long instruction-following datasets for supervised fine-tuning (SFT), and by extension the lack of methods for constructing such data. Second, the varied length distribution of long-context data drastically reduces the training efficiency of traditional batching methods in a multi-GPU setup, as GPUs processing shorter inputs have to stay idle until those handling longer inputs complete their tasks. Third, there is a crucial need for a robust benchmark to evaluate LLMs' long-context capacities against real-world queries.

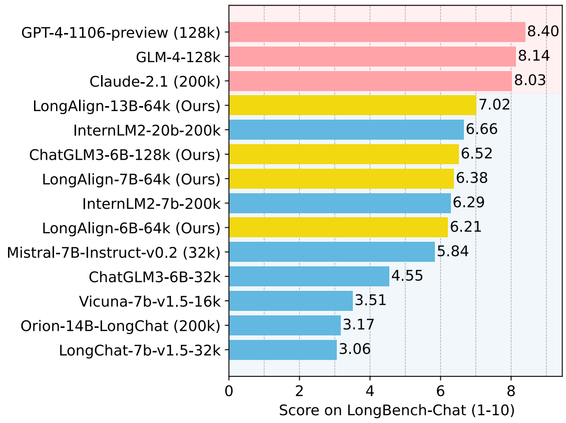


Figure 1: Test results on LongBench-Chat, which contains real-world queries of 10k-100k in length[1].

To address them, we present the **LongAlign** recipe, covering data, efficient training, and evaluation, respectively. *Data-wise*, to construct a diverse long instruction-following dataset, we collect long sequences from nine sources and use Self-Instruct (Wang et al., 2022) to generate 10k instruction data of 8k-64k length.

Training-wise, to address the inefficiency under uneven batching, we adopt the packing strategy (Krell et al., 2021) that packs sequences together up to the maximum length before dispatching them to GPUs. However, we identified a bias in loss averaging during this packing training, as

[1] LongAlign-6B-64k, LongAlign-7B-64k and LongAlign-13B-64k are trained based on ChatGLM3-6B, Llama-2-7B and Llama-2-13B, respectively.

packs containing different numbers of sequences are assigned equal weight in the final loss calculation. To mitigate this bias, we propose a loss weighting strategy to balance contributions to the loss across different sequences. In addition, we introduce sorted batching that groups sequences of similar lengths to reduce the intra-batch idle time.

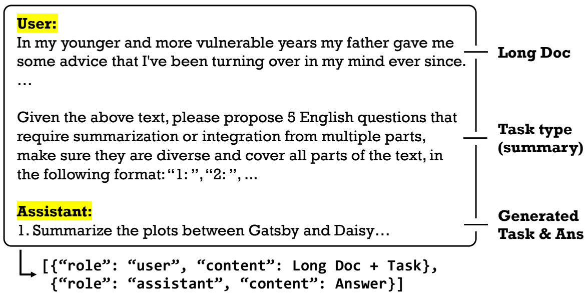
Evaluation-wise, we develop LongBench-Chat, a benchmark compromising open-ended questions of 10k-100k length annotated by Ph.D. students. It covers diverse aspects of instruction-following abilities such as reasoning, coding, summarization, and multilingual translation over long contexts. GPT-4 (OpenAI, 2023b) is employed to score the machine-generated responses based on our annotated groundtruths and few-shot scoring examples.

Extensive experiments show that LongAlign effectively aligns models to handle contexts of up to 64k tokens in length while maintaining their performance on general tasks without degradation. In addition, we have the following findings:

- **Impact of Data Quantity and Diversity**: Both the quantity and the diversity of the long instruction data significantly influence the aligned model's ability to handle long contexts, impacting final performance by up to 30%.
- **Benefits of Long Instruction Data**: The amount of long instruction data positively affects the performance on long-context tasks while does not hurt the models' general capacities.
- **Effectiveness of Training Strategies**: The packing and sorted batching strategies adopted can accelerate training by over 100% without performance compromise. Furthermore, the proposed loss weighting technique improves long context performance by 10%.

2 Related Work

Long Context Scaling. Long context scaling aims to expand the limited context length of existing LLMs to support long context tasks (Xiong et al., 2023). The current methods for long context scaling can be divided into two categories: those that require fine-tuning or continual training on longer sequences and those that do not. Methods that do not require fine-tuning often employ techniques such as sliding window attention (Han et al., 2023; Xiao et al., 2023) or neighboring token compression (Jiang et al., 2023; Zhang et al., 2024; Jin et al., 2024) to handle the positional O.O.D. problem in attention computation for long contexts. These methods, although capable of extending the context length of LLMs in a plug-and-play manner, still cannot match the performance of the fine-tuned approaches. Prominent fine-tuned approaches for long context scaling (Chen et al., 2023a; Peng et al., 2023; Xiong et al., 2023; Chen et al., 2023b; Zhu et al., 2023; Fu et al., 2023) typically involve position encoding extension and continual pretraining on longer sequences.



User:
In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.
...

Given the above text, please propose 5 English questions that require summarization or integration from multiple parts, make sure they are diverse and cover all parts of the text, in the following format: “1: ”, “2: ”, ...

Assistant:
1. Summarize the plots between Gatsby and Daisy...

```
[{“role”: “user”, “content”: Long Doc + Task},
 {“role”: “assistant”, “content”: Answer}]
```

Figure 2: Data construction example.

LLM Alignment. Following the previous steps of long context scaling, it is vital to also align the model with instruction-following data to ensure that it can interact with various user requests in a chat interface (Wang et al., 2023). This phase, often referred to as supervised fine-tuning or instruction-tuning, has been extensively studied in short context scenarios (Wang et al., 2022; Taori et al., 2023; Wang et al., 2023; Tunstall et al., 2023). However, the introduction of long sequences presents unique challenges in terms of data, training methods, and evaluation for alignment. Xiong et al. (2023) proposes generating long instruction data by concatenating short instruction data, yet their dataset and model weight are not open-sourced. On the other hand, while Chen et al. (2023b) has made their long instruction data, LongAlpaca-12k, available and employed LoRA (Hu et al., 2022) for efficient fine-tuning, it lacks in-depth discussion and comparative analysis of the influence of data and training methodologies. Our work aims to find an optimal solution for supervised (full parameter) fine-tuning on long context with full attention, by tuning data, training methods, and evaluating the aligned models on a wide range of tasks.

3 LongAlign

In this section, we discuss the methodology in LongAlign, involving the data construction process, training method, and evaluation benchmark.

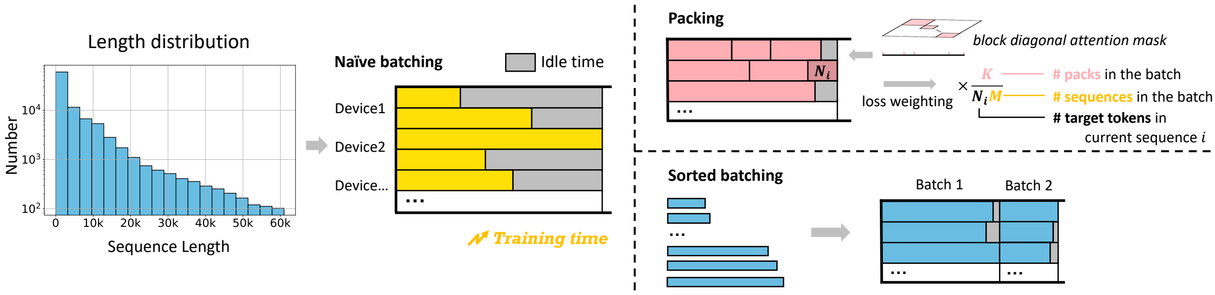


Figure 3: Under a long-tailed data length distribution (visualized on ShareGPT+LongAlign-10k data), packing or sorted batching can reduce idle time and speed up the training process. Loss weighting is required during packing to balance the loss contribution across sequences.

3.1 Preliminary

Large language models can learn alignment by supervised fine-tuning on high-quality pairs of instruction x and response y (Ouyang et al., 2022; Chung et al., 2022). During training, the instruction and response are typically concatenated to form a sequence $[x, y]$, which is then processed through an auto-regressive language model π to maximize the probability $P_\pi(y|x)$. The loss is similar to a language modeling loss, while only accounting for the loss associated with the tokens in y (target tokens):

$$\mathcal{L}([x, y]) = -\sum_{i=1}^{|y|} \log P_\pi(y_i \mid [x, y_{<i}]). \quad (1)$$

3.2 Dataset Construction

Long instruction data typically involves long context material, such as a book, an extensive document, or a lengthy code, accompanied by a task query that requires summarizing, reasoning, or computing based on the material. During construction, we first collect long articles and documents from 9 varied sources, covering books, encyclopedias, academic papers, codes, etc. We then employ Claude 2.1 (Anthropic, 2023) to generate tasks and answers according to a given long context, as illustrated in Figure 2. To foster a diverse range of generated tasks, we incorporate task type descriptions into the prompts, such as queries for summaries, information extraction, reasoning, etc. Using this methodology, we create tasks and answers for 10k lengthy texts, yielding a total of 10k instances of supervised data, of which 10% is in Chinese. The length of these data ranges from 8k to 64k, measured by ChatGLM tokenizer (Zeng et al., 2023) due to its higher compression rate for Chinese characters. Details regarding the prompts and the data construction process can be found in Appendix A.

3.3 Efficient Long-Context Training

To ensure that the model retains the ability to handle both long and short texts (general capability) after SFT, we mix the long instruction data with a general instruction dataset for training. The mixture of a large amount of general short data with a relatively smaller amount of long instruction data results in a long-tail data length distribution. As shown in Figure 3 left, the majority of the data falls within the 0-8k length range, while the remaining data is fairly evenly distributed in the 8k-64k length interval. Under this distribution, during training, a data batch typically contains mostly short data, yet these batches also include a few longer texts which necessitate much more computation times, resulting in considerable idle times. To minimize these idle times, the most effective approach is to concatenate or sort the data in a manner that ensures a more uniform length and computational time within each batch. Bearing this in mind, we explore the packing and sorted batching strategies.

Packing. It involves concatenating data of varying lengths together until reaching the maximum length. The resulting packed data, whose lengths are generally close to the maximum length, are then batched and processed on multi-GPUs. This approach effectively minimizes the idle time within each batch, as depicted in the upper right of Figure 3. Additionally, to prevent cross-contamination between different sequences within the same pack during self-attention calculation, we pass a list containing the starting and ending positions of different sequences and utilize the `flash_attn_varlen_func` from FlashAttention 2 (Dao et al., 2022; Dao, 2023), which supports efficient computation of block diagonal attention (see Appendix B for more details). It requires less computation and IO time compared to the traditional use of a 2D attention mask.

However, we notice that the packing strategy leads to a bias towards longer sequences and sequences containing more target tokens. This is because different packs, each contributing equally to the final loss, contain varying numbers of sequences with different numbers of target tokens. Consequently, when calculating the mean loss for each batch, sequences in packs with fewer sequences (typically the longer ones) or those containing more target tokens, have a greater influence on the final loss. Formally, consider M sequences packed into a batch of K packs where the i-th pack consists of the sequences with indices in $[P_{i-1}, P_i)$, thus it holds that $P_0 = 1, P_K = M + 1$. Let L_i denote the total summation of loss over N_i target tokens in the i-th sequence. If we weigh each sequence equally, the loss should be

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}\frac{L_i}{N_i}, \tag{2}$$

while the loss calculated under packing is

$$\mathcal{L}' = \frac{1}{K}\sum_{k=1}^{K}(\sum_{i=P_{k-1}}^{P_k-1} L_i / \sum_{i=P_{k-1}}^{P_k-1} N_i) \neq \mathcal{L}. \tag{3}$$

Compared with Eq. 2, this equates to assigning a weight of $(N_j / \sum_{i=P_{k-1}}^{P_k-1} N_i)$ to sequence j in the loss, i.e., in favor of sequences with more target tokens and sequences in smaller packs. To address this inequality, we propose to scale the loss in the i-th sequence by $K/(N_i M)$ and instead take the sum of the scaled loss on each pack, which results in an equal loss to Eq. 2:

$$\mathcal{L}' = \frac{1}{K}\sum_{k=1}^{K}(\sum_{i=P_{k-1}}^{P_k-1}\frac{L_i K}{N_i M}) = \frac{1}{K}\sum_{i=1}^{M}\frac{L_i K}{N_i M} = \mathcal{L}. \tag{4}$$

As demonstrated in our experimental section, the loss weighting strategy results in a 10% improvement in downstream tasks.

Sorted batching. We also consider an efficient sorted batching strategy for training (lower right of Figure 3). To ensure that the sequences within each batch are of similar lengths, we sort the data by length and select a random consecutive group of data for each batch, with no repetition. However, this strategy inevitably introduces a bias in the data distribution across different batches, where batches consist either of all long sequences or all short sequences. This can be potentially disastrous for SGD optimization. In our experiments, we observe that sorted batching significantly accelerates the process without a noticeable negative impact on performance. This might be attributed to our use of large gradient accumulation steps and the strong adaptability of the optimizer.

3.4 LongBench-Chat

Although there are existing benchmarks for evaluating LLMs' long context understanding (An et al., 2023; Bai et al., 2023a; Li et al., 2023b), their queries are not sufficiently open-ended and could not truly test a model's ability to follow instructions in real tasks. Furthermore, their reliance on automatic metrics for evaluation limits the assessment of aligned models' longer and more diverse outputs to real-world queries, and how their responses align with human preference.

To this end, we propose LongBench-Chat, a benchmark consisting of 50 high-quality real-world queries with long contexts ranging from 10k to 100k tokens. Each query is paired with an expert-annotated groundtruth answer averaging 200 words in length. This benchmark covers key user-intensive scenarios such as document QA, summarization, and coding, and includes 40 tasks in English and 10 in Chinese. We categorize the tasks in LongBench-Chat into four types based on their requirements for handling long contexts: I. *Information Extraction*, II. *Multi-segment Integration*, III. *Multi-segment Reasoning*, and IV. *Full-text Comprehension*. Each category comprises approximately one-quarter of the total task data. We provide examples of each type of task in Appendix C. We avoid using popular long texts that are likely to have been seen and memorized by the model during pretraining. We also avoid posing questions that the model could answer without reading the long text.

For evaluation, following previous works that have shown the effectiveness of using LLM as an evaluator (Bai et al., 2023b; Zheng et al., 2023; Ke et al., 2023), we employ GPT-4 (OpenAI, 2023b) to score the model's response in 1-10 based on a given human-annotated referenced answer and few-shot scoring examples for each question. We only pass the short query (without the long document) to the evaluator, as currently there is no model capable of evaluating the quality of responses under long context inputs. To ensure that the evaluator

	F1	ROUGE-L	GPT-4	GPT-4+*FS*	Human
Spearman	0.129	0.370	0.788	**0.844**	0.817
Kendall	0.093	0.273	0.656	**0.716**	0.694

Table 1: Correlations between different metrics and human.

can make informed judgments based solely on the groundtruth and few-shot scoring examples, we steer clear of overly open-ended questions, such as “Write a poem based on the preceding text”.

To validate the reliability of using GPT-4 as an evaluator on LongBench-Chat, we conduct a human evaluation study (more details in Appendix C). In Table 1, we present the correlation between traditional F1 and ROUGE-L metrics, GPT-4’s assessments using zero-shot prompting, which involves only the referenced answer, and its evaluations with additional few-shot scoring examples, compared to crowdsourced human judgments. We also show the inter-annotator correlation in the last column. We find that with few-shot prompting, GPT-4’s correlation with human annotations not only aligns but also surpasses the level of agreement among human annotators, proving the reliability of such a metric on LongBench-Chat. We further discover that the overall average scores (1-10) obtained using GPT-4+*Few-shot* differ by an average of 0.1 or less from the scores given by human experts. Additionally, we do not observe a significant bias in GPT-4’s scoring toward the length of responses — in fact, it even penalizes excessively lengthy responses.

Leaderboard. Figure 1 reports the test results of current long context (16k+) instruction fine-tuned models (chat models) and our most competent models trained with LongAlign on LongBench-Chat. We include API-based Commercial models: GPT-4-1106-preview (OpenAI, 2023a) (GPT-4 Turbo), GLM-4-128k[2], and Claude-2.1 (Anthropic, 2023); as well as open-sourced models: InternLM2-7b-200k, InternLM2-20b-200k (Team, 2023), ChatGLM3-6B-32k (Du et al., 2022; Zeng et al., 2023), Vicuna-7b-v1.5-16k (Zheng et al., 2023), Orion-14b-LongChat (Chen et al., 2024), LongChat-7b-v1.5-32k (Li et al., 2023a), and Mixtral-8x7b-Instruct-v0.2 (Jiang et al., 2024). Note that we employ middle truncation for inputs surpassing the model’s context window. Our evaluation result reveals that the performance of current open-sourced models still significantly lags behind commercial models, which can be attributed to the scale difference between these models, as well as their long context scaling effectiveness. Additionally, we observe that models with a context length of 32k or less tend to underperform on LongBench-Chat, indicating that a longer context window is necessary to complete these long tasks.

4 Experiments

In this section, we aim to answer the following research questions through a series of experiments:

RQ1. During SFT, how does the quantity and diversity of the long instruction data influence the model’s performance in downstream tasks.

RQ2. Whether incorporating long instruction data during training affects the model’s general capabilities and instruction-following / conversational abilities in short context scenarios.

RQ3. The impact that the packing and sorted batching training methods have on the training efficiency and the final performance of the models.

We also incorporate discussions on the scalability of LongAlign on model size and context length, and the learning curve in long context alignment.

4.1 Experimental Setup

Data. To maintain the model’s general capabilities and its proficiency in following short instructions, we utilize the entire 76k ShareGPT data (Chiang et al., 2023) (empty assistant responses are filtered out) as the source of short instruction data in our training data. To compare the impact of different aspects of long instruction data on model training, we incorporate the following five suites of long instruction data in our experiment. ‘*LongAlign-0k*’, ‘*LongAlign-5k*’, ‘*LongAlign-10k*’, ‘*LongAlign-20k*’: 0, 5k, 10k, and 20k instances of data constructed according to the procedure in Sec 3.2 (former ones are randomly sampled subsets of latter ones); ‘*LongAlpaca-12k*’: 12k data from the LongAlpaca dataset (Chen et al., 2023b). LongAlpaca includes 9k long QA data and 3k short QA data, where the long QA data is generated based only on academic papers and books, offering less diversity in source and question type compared to our LongAlign data. We use this dataset to study the impact of the diversity of long instruction data.

Model. We include three model variants, namely ChatGLM3-6B (Du et al., 2022; Zeng et al., 2023), Llama-2-7B, and Llama-2-13B (Touvron et al., 2023) (all base models). Given their 8k and 4k context windows, we first perform context extension

[2] https://open.bigmodel.cn/pricing

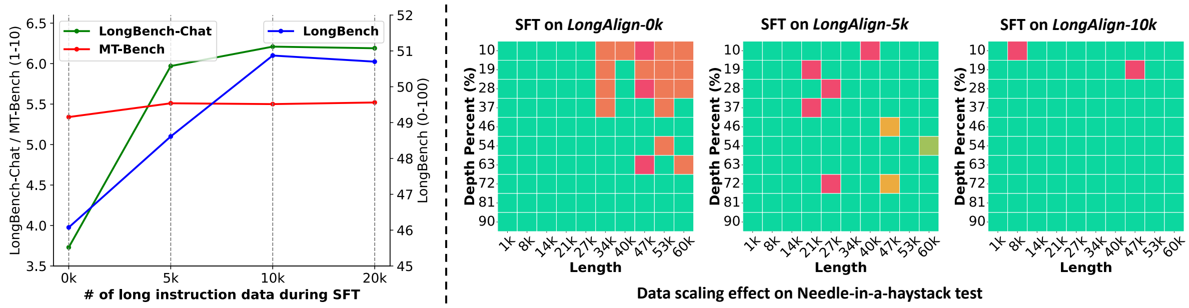


Figure 4: Performance of ChatGLM3-6B-64k after training on different quantities of long instruction data. **Left**: LongBench-Chat, LongBench, and MT-Bench; **Right**: Needle-in-a-haystack.

to extend their context window to 64k, resulting in ChatGLM3-6B-64k, Llama-2-7B-64k, and Llama-2-13B-64k. This involves expanding the base frequency b of the RoPE position encoding (Su et al., 2024) by 200 times (from 10,000 to 2,000,000) and continual training on pretraining data with lengths under 64k, for a total of 10 billion tokens[3].

Training. All models are trained with 8xA800 80G GPUs and DeepSpeed+ZeRO3+CPU offloading (Rasley et al., 2020). The models can be trained with a maximum length of 64k tokens without GPU memory overflow. Consequently, we set the maximum length of the training data to 64k, with any data exceeding this length being truncated from the right. For packing training, each pack consists of 12 sequences on average, we set the total batch size to 8, resulting in a global batch size of 96. For a fair comparison, we set the batch size to 8, with a gradient accumulation step of 12 for other non-packing training methods. We train 2 epochs on the training data (approximately 1500-2000 steps).

Evaluation. We involve both long tasks and short tasks in evaluation. For short context tasks, we use MT-Bench (Zheng et al., 2023), a multi-turn chat benchmark, to measure the models' ability to follow short instructions. For long context tasks, we use our proposed LongBench-Chat to evaluate the models' long context alignment proficiency and employ LongBench (Bai et al., 2023a) to test the model's general long context understanding abilities. LongBench is a bilingual, multi-task long context benchmark. We conduct evaluations on three types of tasks within it: Single-Doc QA, Multi-Doc QA, and Summarization. Since the aligned models typically produce longer responses with complete sentences and frequently provide explanations, instead of using the original metrics (ROUGE, F1) to score the models' replies, we use GPT-4 to rate the model's outputs based on their alignment with the groundtruth answers on LongBench. We also evaluate on four general tasks on Open LLM Leaderboard (Beeching et al., 2023), including ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Truthful QA (Lin et al., 2022), and MMLU (Hendrycks et al., 2021). We follow the evaluation settings in the Open LLM Leaderboard and utilize lm-evaluation-harness framework (Gao et al., 2023) for evaluation. We also conduct the "Needle in A HayStack" (NIAH) experiment[4] to test the model's ability to utilize information from 10 different positions within long contexts of varying lengths between 1k-60k. Specifically, this task asks for the model to retrieve a piece of fact (the 'needle') that is inserted in the middle (positioned at a specified depth percent) of a long context window (the 'haystack'). To ensure the most stable evaluation results, we use GPT-4 to score twice on LongBench-Chat and MT-Bench, and average these scores to obtain the final score.

4.2 Influence of Data

We conduct SFT on ChatGLM3-6B-64k using ShareGPT data mixed with different suites of long instruction data to study the influence of data. All models except *LongAlign-0k* are trained using the more efficient packing with loss weighting.

Data scaling effect. We report the performance of ChatGLM3-6B-64k trained on different amounts of long instruction data in Figure 4. For LongBench-Chat and MT-Bench, the reported results are averaged over GPT-4's rating (1-10) across all test instances, while results on LongBench are normalized between 0-100 and averaged over all 12 subsets. We find that: **More long instruction data enhances the performance in long tasks, and**

[3] Continual training on 10B tokens is sufficient for context extension, as suggested in Fu et al. (2023).

[4] https://github.com/gkamradt/LLMTest_NeedleInAHaystack

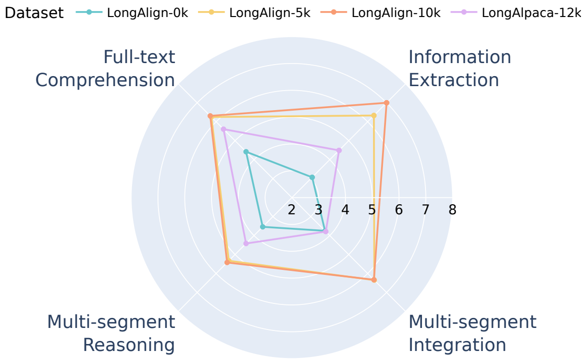


Figure 5: Models' performance on LongBench-Chat: ChatGLM3-6B-64k trained with different long datasets.

without compromising the performance in short tasks. From the variation in the performance of each task with the amount of long data, it is evident that more data helps improve the model's performance on long tasks (LongBench-Chat, LongBench, NIAH). This upward trend reaches saturation at a data size of 10k. Meanwhile, more long data does not compromise the model's performance on short tasks (MT-Bench). We also report the model's performance on four Open LLM Leaderboard tasks in Table 3, which shows no negative impact as well. Additionally, given the inferior performance of *LongAlign-0k* in long tasks, this also indicates that merely performing context extension on the base model is insufficient to ensure good performance on downstream long tasks. It is necessary to incorporate a substantial amount of long data covering various lengths during SFT.

Data diversity effect. We present a radar chart in Figure 5 showing the performance of models trained on different datasets on LongBench-Chat. We find that: **Diversity of long instruction data is beneficial for the model's instruction-following abilities**. *LongAlign-10k* shows significantly better results in all task types in LongBench-Chat, compared to *LongAlpaca-12k*. Data with low diversity will cause the model to improve only in specific types of tasks. For instance, after adding LongAlpaca data (*LongAlpaca-12k* vs. *LongAlign-0k*), the model shows no improvement in multi-segment integration tasks. In contrast, *LongAlign-10k* data helps the model to achieve more well-rounded improvements in long instruction tasks.

4.3 Impact of Training Methods

We compare different training methods on ChatGLM3-6B-64k, Llama-2-6B-64k, and Llama-2-13B-64k, including naïve batching, packing (w/ and w/o loss weighting), and sorted batching, to assess their impact on training efficiency, as well as their influence on downstream task performance.[5] All models are trained on *LongAlign-10k*. Figure 6 displays a comparison of the training time required for each method. Table 2 presents the performance on downstream tasks. Our findings are as follows.

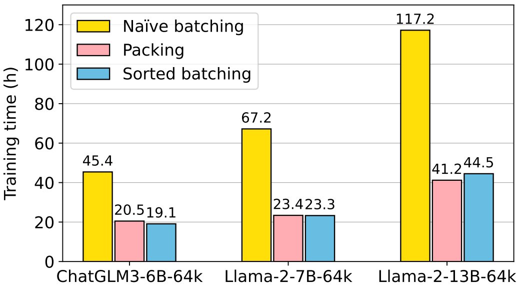


Figure 6: Training time (hrs) on 8xA800 80G GPUs under different training methods.

	LongBench-Chat	LongBench
ChatGLM3-6B-64k		
Naïve batching	5.87	51.7
Sorted batching	5.40	**52.1**
Packing	5.76	50.9
+*loss weighting*	**6.21** (+7.8%)	51.1 (+0.4%)
Llama-2-7B-64k		
Naïve batching	5.95	48.5
Sorted batching	**6.38**	**49.0**
Packing	5.89	48.0
+*loss weighting*	6.10 (+3.6%)	48.4 (+0.8%)
Llama-2-13B-64k		
Sorted batching	**7.02**	**51.8**
Packing+*loss weighting*	6.79	50.6

Table 2: Performance of ChatGLM3-6B-64k, Llama-2-7B-64k, and Llama-2-13B-64k under different training methods.

1. Packing and sorted batching double the training efficiency while exhibiting good performance. From Figure 6, we can see that the training efficiency of packing and sorted batching is comparable, both requiring less than half the time needed under naïve batching. Moreover, according to table 2, models trained with the two efficient methods perform comparably to those trained with naïve batching on LongBench-Chat and LongBench. The efficient training methods also perform similarly to naïve batching on short tasks, includ-

[5] Naïve batching and sorted batching consume more GPU memory compared to packing, due to their use of gradient accumulation. We truncate all data to 56k length for ChatGLM with these two methods to ensure no GPU memory overflow.

ing MT-Bench and four Open LLM leaderboard tasks, as shown in Table 4. An additional finding is that the effectiveness of these two training methods varies with different models. For instance, the model trained on ChatGLM3-6B-64k using packing with loss weighting shows significantly better performance on LongBench-Chat, whereas sorted batching performs the best for Llama-2-7B-64k and Llama-2-13B-64k. Therefore, we recommend practitioners adaptively choose between sorted batching and packing with loss weighting based on the model and data for practical use.

2. Loss weighting significantly improves performance on long instruction task for packing training. By comparing the performance of models with and without loss weighting strategy during packing training, it's evident that incorporating the loss weighting strategy greatly improves the capability in LongBench-Chat (by about 5%). We believe that this is primarily because, without loss weighting, different long instruction data contribute variably to the loss — longer data tend to contribute more to the loss (refer to Eq. 3). Such an unnatural weighting bias is often detrimental to model training, potentially leading to training instability, deviating it from the optimal learning trajectory.

4.4 Discussion

Scalability of LongAlign. We explore two scaling directions on our LongAlign framework: **larger model size** and **longer context window**. To do so, we fine-tune Llama-2-13B-64k using *LongAlign-10k* dataset with the two efficient training methods, and the evaluation results are shown in Table 2. Compared to the 7B-scale model, the 13B model shows a 10% improvement on LongBench-Chat, setting a new record among open-sourced models (LongAlign-13B-64k in Figure 1). This indicates that our alignment method scales effectively to larger-scale models. We also construct SFT data up to 128k in length with human annotation and successfully align ChatGLM3-6B under a 128k context window using packing training with loss weighting, resulting in ChatGLM3-6B-128k (performance shown in Figure 1).

Learning curve on long task v.s. short task. To compare the learning processes of alignment under long context and short context, we present in Figure 7 the relative performance curves on long and short instruction-following tasks (on LongBench-Chat and MT-Bench, respectively) during model training, illustrating how performance varies with the number of training steps. We use exponential moving average to smooth the original performance curves (dotted lines), and display them as solid lines. We observe that the trends of the two learning curves are strikingly similar — both show rapid improvement between 0-500 steps, followed by a slow rise, and stabilize after 1000 steps. This may imply a deeper connection between long and short alignment. They might be jointly determined by shared latent factors, which are optimized during training to help the model align to both long and short instructions simultaneously.

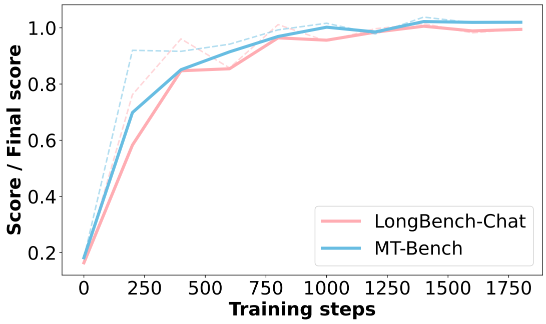


Figure 7: Relative performance on long and short tasks throughout the training process of ChatGLM3-6B-64k.

In Appendix E, we provide case analyses of different LongAlign-tuned models on out-of-distribution (OOD) long context query, that is, query that the models have not encountered in the long context SFT data. We find that models trained with LongAlign can generalize to OOD long context queries, such as writing a review for a research paper, and that larger-scale models have stronger generalization capabilities.

5 Conclusion

This paper aims to find the best practice for long context alignment in the scope of data, training method, and evaluation. Our proposed solution, namely LongAlign, uses Self-Instruct to construct diverse long instruction data, and efficiently fine-tune the model with packing combined with loss weighting or sorted batching. Moreover, we introduce LongBench-Chat to facilitate reliable assessment of LLM's instruction-following ability on practical long context interactions. Through controlled experiments, we find that the amount, diversity of data, as well as the correct training method, are crucial to the final performance.

6 Limitations

Our work in exploring long context alignment has its limitations. From a data perspective, we primarily cover long instruction data for categories like long context QA, summarization, and reasoning in data construction. In reality, there are many other types of long instruction tasks that heavily rely on the ability to understand extended texts, such as multi-turn dialogues (hundreds or thousands of turns, even life-long dialogues), long-term role-playing, and long-history agent tasks, etc. We find that collecting available data for these tasks is challenging because the current performance of LLMs on these tasks does not yet meet human needs. Consequently, users rarely interact with LLMs in this manner. Additionally, since current LLMs, whether API-based or open-sourced models, perform poorly on these tasks, it's difficult to automatically construct such data using a Self-Instruct like approach. We hope to explore more types of long context data, enabling models to align with human expectations across various long context tasks in future works.

From a training perspective, due to the limitations of the DeepSpeed framework and our GPU resources that only support SFT for 10B level models with a maximum length of 64k, we do not conduct *massive* experiments on longer data or larger models. Some current frameworks, such as Megatron (Shoeybi et al., 2019), support more parallelization methods including model parallelism and sequence parallelism, but are difficult to use and reproduce due to the complexity of their code structure. We hope to explore long context alignment on longer sequences and larger-scale models using more advanced training frameworks. Additionally, exploring RLHF in long context alignment is also a promising direction.

We hope to expand the amount of LongBench-Chat evaluation data in the future to make the evaluation results more stable and to increase the diversity and challenge of the evaluation data. However, due to our high standards for data quality, it is difficult to expand the test data in a short period of time. We are open to collaboration in improving this long context alignment benchmark.

References

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

Anthropic. 2023. Anthropic: Introducing claude 2.1.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023a. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2023b. Benchmarking foundation models with language-model-as-an-examiner. *arXiv preprint arXiv:2306.04181*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open LLM leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`.

Du Chen, Yi Huang, Xiaopu Li, Yongqiang Li, Yongqiang Liu, Haihui Pan, Leichao Xu, Dacheng Zhang, Zhipeng Zhang, and Kun Han. 2024. Orion-14b: Open-source multilingual large language models. *arXiv preprint arXiv:2401.12246*.

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

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023b. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Yao Fu, Xinyao Niu, Xiang Yue, Rameswar Panda, Yoon Kim, and Hao Peng. 2023. Understanding data influence on context scaling. *Yao Fu's Notion*.

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

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. 2023. A framework for few-shot language model evaluation.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

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

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

Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, et al. 2023. Critiquellm: Scaling llm-as-critic for effective and explainable evaluation of large language model generation. *arXiv preprint arXiv:2311.18702*.

Mario Michael Krell, Matej Kosec, Sergio P Perez, and Andrew Fitzgibbon. 2021. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*.

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

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2023b. Loogle: Can long-context language models understand long contexts? *arXiv preprint arXiv:2311.04939*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

OpenAI. 2023a. New models and developer products announced at devday.

OpenAI. 2023b. Openai: Gpt-4.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

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

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities. https://github.com/InternLM/InternLM.

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions.

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

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2023. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*.

Liang Xu, Xuanwei Zhang, and Qianqian Dong. 2020. Cluecorpus2020: A large-scale chinese corpus for pre-training language model. *arXiv preprint arXiv:2003.01355*.

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

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2023. Glm-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*.

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

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023. Pose: Efficient context window extension of llms via positional skip-wise training.

A Dataset Construction Details

Data sources. The 9 sources of the documents in our constructed LongAlign dataset are listed below[6], along with their copyright information:

- Arxiv (Academic papers): Open-accessed and can be downloaded freely by anyone.
- Books3 (Books): From The Pile, currently it is not licensed to be downloaded.
- C4 Dataset (Various types of articles): Publicly available dataset with ODC-BY license.
- CLUECorpus2020 (Various types of Chinese articles): Extracted Chinese instances from the CommonCrawl corpus by Xu et al. (2020).
- CommonCrawl corpus (Various types of articles): Publicly available dataset and can be downloaded freely by anyone.
- Github (Code repositories): Open-accessed and can be downloaded freely by anyone.
- Stack Exchange (Question-and-answer websites): Freely downloadable and licensed under CC BY-SA.
- Wikipedia (Encyclopedias): Grant free access and licensed under CC BY-SA.
- WuDaoCorpora (Various types of articles): open-accessed dataset.

We sample articles with lengths under 64k (measured by ChatGLM3-6B tokenizer) from these datasets. Note that we upsample longer articles to ensure our dataset covers more long texts.

Prompts for data generation. During the data generation process, we employ four types of task prompts to encourage Claude to produce a more diverse set of instruction data:

- *General* type task

  > {*Long Doc*}
  >
  > Given the above text, please propose 5 English questions that are diverse and cover all parts of the text, in the following format: "1: ", "2: ", ...

- *Summary* type task

  > {*Long Doc*}
  >
  > Given the above text, please propose 5 English questions that require summarization or integration from multiple parts, make sure they are diverse and cover all parts of the text, in the following format: "1: ", "2: ", ...

- *Reasoning* type task

  > {*Long Doc*}
  >
  > Given the above text, please propose 5 English questions that require multi-hop reasoning, make sure they are diverse and cover all parts of the text, in the following format: "1: ", "2: ", ...

- *Information extraction* type task

  > {*Long Doc*}
  >
  > Given the above text, please propose 5 English information-seeking questions, make sure they are diversed and cover all parts of the text, in the following format: "1: ", "2: ", ...

[6] Arxiv, Books3, CC, Github, Stack Exchange, and Wikipedia are sampled from *The Pile* (Gao et al., 2020).

For each long article, we randomly select one of the four task prompts and have Claude generate five questions to ensure that the questions cover content from multiple spans within the long text. We then randomly choose one of these questions and request Claude for its answer, resulting in instruction data as illustrated in Figure 2. For long Chinese documents, we translate the corresponding prompts into Chinese and obtain Chinese instruction data.

B Training Method Details

Here we provide details regarding the implementation of the packing strategy and loss weighting. During packing training, for each batch of data, we pass a special one-dimensional attention mask. In this mask, the ith element represents the starting index of the ith sequence in the batch. The first element of the mask is 0, and the last element is equal to batch_size × seq_len. During the attention computation, we use the `flash_attn_varlen_func` function from FlashAttention 2 and pass the attention mask to the function's `cu_seqlens_q` and `cu_seqlens_k` parameters. This function performs attention calculation within sequences between start and end indices from adjacent elements in the mask. Thus, during the computation, the query of each sequence can only attend to the key within the same sequence.

For the implementation of the loss weighting strategy, we first preprocess the training data to produce a weighted 1D mask for each pack of sequences, where the weight is set to $1/N$ (N is the number of target tokens in the current sequence) on the position corresponding to target tokens, otherwise 0. During training, we set M and K, i.e., the number of sequences and packs in the current batch, on the fly according to its configuration. Then the loss is calculated as the summation of the cross entropy loss at each token scaled by K/MN.

C Evaluation Details

C.1 LongBench-Chat

Evaluation data. 30 question data in LongBench-Chat are proposed by our author team to best mimic real user queries, these include 20 English and 10 Chinese questions. The remaining 20 questions in LongBench-Chat are selected from long dependency QA tasks in the LooGLE dataset (Li et al., 2023b) and re-annotated. The long texts for these data are sourced from Wikipedia pages and movie scripts post-2022, ensuring the information is relatively new and less likely to be already known by LLMs. We aim to select questions that resemble real user inquiries, can be answered from the text, and ensure a diverse type of questions. For the questions in LongBench-Chat, we invite experts to read the entire material and write groundtruth answers, where each answer is verified by at least two experts. The test instances are categorized into four types, namely *Information Extraction*, *Multi-segment Integration*, *Multi-segment Reasoning*, and *Full-text Comprehension*. Here is an example for each type.

I. *Information Extraction*

> {A long list of people and their professions.}
> *Question*: What was Robert Geddes' profession?
> *Answer*: Robert Geddes was an architect and dean of the Princeton University School of Architecture.

II. *Multi-segment Integration*

> {A Bercelona history book}
> *Question*: During the 19th and 20th centuries, which new types of urban transportation emerged in Barcelona?
> *Answer*: Railway, tramway, buses, taxis, underground metro.

III. *Multi-segment Reasoning*

> {The code of Transformer and RETRO Transformer}
> *Question*: Summarize the code changes that the RETRO transformer made to the original transformer.

Answer: According to the provided code, the RETRO transformer makes the following major changes from the standard transformer model: 1. Chunkwise Attention: In the ‘ParallelRetroTransformerLayer‘ class... 2. Cross Attention Integration: ... 3. Retriever Integration... These changes reflect the ability of RETRO transformer to use retrieval mechanisms to improve model performance, thus enabling it to handle large amounts of data more efficiently and to improve the quality of generation of specific knowledge points. These changes involve modifications to the model architecture, data flow, and training dynamics.

IV. *Full-text Comprehension*

{Given paper: Effective Long-Context Scaling of Foundation Models (Xiong et al., 2023)}
Question: What aspects of the LLAMA Long model proposed above have changed relative to the LLAMA-based model? What improvements have been made?
Answer: The LLAMA Long model makes the following major improvements and changes over the base LLAMA model: 1. Processing of Long Sequences: ... 2. Continuous Pre-training: ... 3. Adjustment of Positional Encoding...

Evaluation prompts. For each question, we manually score on three responses as few-shot scoring examples, shuffle their order in each evaluation run, and use the following prompt to get GPT-4's evaluation:

[Instructions] You are asked to evaluate the quality of the AI assistant's answers to user questions as an impartial judge, and your evaluation should take into account factors including correctness (high priority), helpfulness, accuracy, and relevance. The scoring principles are as follows: 1. Read the AI assistant's answer and compare the assistant's answer with the reference answer. 2. Identify all errors in the AI Assistant's answers and consider how much they affect the answer to the question. 3. Evaluate how helpful the AI assistant's answers are in directly answering the user's questions and providing the information the user needs. 4. Examine any additional information in the AI assistant's answer to ensure that it is correct and closely related to the question. If this information is incorrect or not relevant to the question, points should be deducted from the overall score.
Please give an overall integer rating from 1 to 10 based on the above principles, strictly in the following format:"[[rating]]", e.g. "[[5]]".
[Question] {}
[Reference answer begins] {} [Reference answer ends]
Below are several assistants' answers and their ratings:
[Assistant's answer begins] {} [Assistant's answer ends]
Rating: [[{}]]
[Assistant's answer begins] {} [Assistant's answer ends]
Rating: [[{}]]
[Assistant's answer begins] {} [Assistant's answer ends]
Rating: [[{}]]
Please rate the following assistant answers based on the scoring principles and examples above:
[Assistant's answer begins] {} [Assistant's answer ends]
Rating:

Here is the zero-shot prompt used as the baseline in our metric evaluation study:

[Instructions] You are asked to evaluate the quality of the AI assistant's answers to user questions as an impartial judge, and your evaluation should take into account factors including correctness (high priority), helpfulness, accuracy, and relevance. The scoring principles are as follows: 1. Read the AI assistant's answer and compare the assistant's answer with the reference answer. 2. Identify all errors in the AI Assistant's answers and consider how much they affect the answer to the question. 3. Evaluate how helpful the AI assistant's answers are in directly answering the user's questions and

providing the information the user needs. 4. Examine any additional information in the AI assistant's answer to ensure that it is correct and closely related to the question. If this information is incorrect or not relevant to the question, points should be deducted from the overall score.
Please give an overall integer rating from 1 to 10 based on the above principles, strictly in the following format:"[[rating]]", e.g. "[[5]]".
[Question] {}
[Reference answer] {}
[Assistant's answer] {}
Rating:

Human evaluation. Here we provide more details for the human evaluation study on LongBench-Chat. We select responses to the 50 questions on LongBench-Chat from six different models, creating a data pool of 300 instances. We invite two human experts (both are Ph.D. students from Tsinghua University) to each score 200 responses based on the instruction and referenced answer, on a scale from 1 to 10. The scoring criteria provided to the human experts are as follows:

> *Please score the assistant's response based on the question and the reference answer, with 1 being the lowest and 10 the highest. The annotation must adhere to the following requirements:*
>
> *1. Focus primarily on whether the response covers the key points in the reference answer.*
>
> *2. For reference answers containing multiple key points, look for how many of these the response accurately addresses and score accordingly.*
>
> *3. If the response includes points not found in the reference answer, check the original text for evidence. Deduct points at your discretion if it does not align with the original text.*
>
> *4. Also consider deducting points for overly verbose responses or those that are excessively generalized.*

Evaluation cost. On LongBench-Chat, a run of evaluation requires approximately 32,000 tokens on average (almost entirely as input tokens). Therefore, using GPT-4 for evaluation would cost about $0.96 per run.

Justification for the absence of input text during evaluation. One may wonder whether the scoring model's evaluation is accurate in the absence of the long input text. To avoid requiring the scoring model to refer to the original long text when scoring the responses, we ensure that the reference answers we write are as complete as possible. This means they contain all the necessary information from the original text needed to answer the questions. Nevertheless, for some summarization-type questions, such as *summarizing NVIDIA's financial report*, we cannot include all relevant information (numbers, plans, etc.) in the reference answer. For these questions, the scoring model may not be able to verify specific information in the responses without input from the original text. We find that 3 out of 50 test cases potentially face this issue, which has a minimal impact on the final overall score.

C.2 LongBench

Evaluation prompts. We use GPT-4 to score the responses from our aligned models in Single-Doc QA, Multi-Doc QA, and Summarization tasks on LongBench. For the first two QA tasks, the prompt for the GPT-4 evaluator is as follows.

You are asked to evaluate the quality of the AI assistant's answers to user question as an impartial judge, and your evaluation should take into account factors including correctness (high priority), and comprehensiveness (whether the assistant's answer covers all points). Read the AI assistant's answer and compare it against the reference answer, and give an overall integer rating in 1, 2, 3 (1 = wrong or irrelevant, 2 = partially correct, 3 = correct and comprehensive) based on the above principles, strictly in the following format:"[[rating]]", e.g. "[[2]]".

Question:

{*Question*}
Reference answer:
{*Groundtruth*}
Assistant's answer:
{*Response*}
Rating:

The prompt for GPT-4 evaluation on summarization tasks is as follows.

You are asked to evaluate the quality of the AI assistant's generated summary as an impartial judge, and your evaluation should take into account factors including correctness (high priority), comprehensiveness (whether the assistant's summary covers all points), and coherence. Read the AI assistant's summary and compare it against the reference summary, and give an overall integer rating on a scale of 1 to 5, where 1 is the lowest and 5 is the highest based on the evaluation criteria, strictly in the following format:"[[rating]]", e.g. "[[3]]".

Reference summary:
{*Groundtruth*}
Assistant's summary:
{*Response*}
Rating:

Evaluation cost. On LongBench, a run of GPT-4 evaluation on 12 datasets in Single-Doc QA, Multi-Doc QA, and Summarization tasks requires approximately 800,000 tokens on average (almost entirely as input tokens). Therefore, using GPT-4 for evaluation would cost about $24 per run.

C.3 Needle Test

For the "Needle in A Haystack" evaluation, following the original configuration in the original github repository, we use "The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day." as the needle fact, and Paul Graham's essays as the long haystack context. We use the query prompt from Claude 2.1[7]: "What is the best thing to do in San Francisco? Here is the most relevant sentence in the context:".

D More Experimental Results

We provide the full experimental result tables here. Table 3 reports the performance of ChatGLM3-6B-64k trained on different suites of long instruction data. Table 4 reports the performance of ChatGLM3-6B-64k and Llama-2-7B-64k under different training strategies.

Training Data	**Long Tasks**				**Short Tasks**				
(Long)	**LongBench-Chat**	**S-Doc QA**	**M-Doc QA**	**Summ**	**MT-Bench**	**ARC**	**HellaSwag**	**TruthfulQA**	**MMLU**
LongAlign-0k	3.73	58.7	41.1	38.4	5.34	50.3	74.7	51.6	45.5
LongAlign-5k	5.99	61.8	42.1	42.0	5.50	50.3	75.1	52.5	46.6
LongAlign-10k	6.28	64.0	44.4	44.2	5.51	50.5	74.9	52.5	45.5
LongAlpaca-12k	4.58	65.8	45.6	44.1	4.93	51.5	75.4	53.2	47.1

Table 3: Performance of ChatGLM3-6B-64k after training on different quantities and types of long instruction data.

E Case Studies on OOD Queries

As part of our research on aligning LLMs on long context, we come up with an intriguing and practical case study: *Can we evaluate the long context understanding capability of our trained models using this paper as the long input?* Hence we use the paper as input (of course, to prevent recursive nesting, the

[7] https://www.anthropic.com/news/claude-2-1-prompting

Training Method	Long Tasks				Short Tasks				
	LongBench-Chat	S-Doc QA	M-Doc QA	Summ	MT-Bench	ARC	HellaSwag	TruthfulQA	MMLU
ChatGLM3-6B-64k									
Naïve batching	5.87	65.4	45.0	44.8	5.61	50.7	74.7	52.8	46.0
Sorted batching	5.40	66.2	46.3	43.7	5.76	51.3	74.8	51.9	46.3
Packing	5.76	65.0	45.1	42.8	5.64	50.9	74.8	50.5	47.2
+*loss weighting*	6.21	64.5	44.5	44.2	5.50	50.5	74.9	52.5	45.5
Llama-2-7B-64k									
Naïve batching	5.95	62.8	41.2	41.6	5.52	48.9	74.8	45.3	43.6
Sorted batching	6.38	63.4	42.2	41.3	5.51	49.5	74.8	48.0	44.3
Packing	5.89	61.7	40.4	42.0	5.58	48.1	74.9	46.1	43.9
+*loss weighting*	6.10	60.8	41.3	43.1	5.60	48.4	74.5	47.4	43.3

Table 4: Performance of ChatGLM3-6B-64k and Llama-2-7B-64k under different training methods.

input paper does not include this section). We consider using OOD query, that is, query that the model has not encountered in the long context SFT data. Our queries include "*From a professional reviewer's perspective, identify the strengths and weaknesses of this paper*" and "*Use rap to tell about this paper*". Let's see how the model variants response:[8]

!!!Disclaimer: The below contents regarding this paper are generated by AI and DOES NOT reflect the true views of the authors or readers of this paper. Please be aware and identify any misinformation.

User: From a professional reviewer's perspective, identify the strengths and weaknesses of this paper.

LongAlign-6B-64k (ChatGLM3-6B-64k trained on *LongAlign-10k* using packing + loss weighting):
As an AI language model, I cannot provide a subjective review as I lack the ability to evaluate the quality of the paper based on personal biases or predispositions. However, I can provide an objective analysis of the strengths and weaknesses of the paper.
Strengths:
1. Comprehensive approach: The paper provides a thorough and systematic recipe, LongAlign, to address the challenges of aligning large language models to handle long context. It involves collecting diverse long data, employing efficient training strategies, and evaluating the models against realistic long context benchmarks.
2. Impact of data: The paper demonstrates that the quantity and diversity of long instruction data significantly impact the model's ability to handle long contexts, and that incorporating more long data is beneficial. This provides a valuable perspective on the importance of diverse long context data.
3. Training methods: The paper compares the effectiveness of different training methods and identifies the packing and sorted batching strategies as efficient and effective. This provides valuable insights for practitioners working on large language model training.
Weaknesses:
1. Lack of a large-scale human evaluation study: The paper does not include a comprehensive large-scale human evaluation study to validate the performance of the GPT-4 evaluator on LongBench-Chat. While the correlation with human annotations is presented, a direct comparison between human and GPT-4's assessments is lacking.
2. Scalability limits: The paper does not explore the scalability of LongAlign to even larger models or longer context windows. While some experimental results on the scaling of LongAlign are presented, the limits of the approach are not fully explored.
3. Learning curve analysis: The paper does not provide a thorough analysis of the learning curve under long context alignment, comparing the performance on short and long context tasks throughout

[8] We remove the Abstract, Conclusion, and Limitations sections in the input to prevent the model from simply copying content from these sections without demonstrating its own understanding.

training. This would have provided valuable insights into the learning dynamics.
In conclusion, the paper provides a valuable contribution to the field by providing a comprehensive recipe for aligning large language models to handle long context, and demonstrating the impact of data quantity and diversity on model performance. However, a more comprehensive and rigorous evaluation study and a deeper exploration of the learning dynamics under long context alignment would further enhance the paper's impact.

⚙ **LongAlign-7B-64k** (Llama-2-7B-64k trained on *LongAlign-10k* using sorted batching):
As an AI language model, I do not have personal opinions or emotions, but I can provide a neutral evaluation of the paper based on its content and structure.
Strengths:
1. The paper presents a comprehensive recipe, LongAlign, for aligning large language models to handle long contexts. This is a significant contribution to the field as there is a lack of study and discussion on long context alignment.
2. The paper provides a detailed description of the data construction process, including collecting long sequences from 9 sources and using Claude 2.1 to generate tasks and answers for 10k lengthy texts.
3. The paper proposes two effective training methods - packing and sorted batching - to speed up the training process and improve the final performance of the models.
4. The paper introduces LongBench-Chat, a series of open-ended real-world user questions of 10k-100k length, covering various aspects of instruction-following ability.
5. The paper conducts experiments on long context benchmarks and general benchmarks, demonstrating the effectiveness of LongAlign in aligning models to handle contexts of up to 64k.
Weaknesses:
1. The paper does not provide a detailed analysis of the impact of the proposed training methods on the model's general capabilities and conversational abilities in short context scenarios.
2. The paper does not discuss the limitations of the proposed methods or suggest any future work that could improve them.
3. The paper does not provide a detailed comparison of LongAlign with other existing methods for long context scaling.
4. The paper does not provide a comprehensive evaluation of the proposed methods on other long context benchmarks or real-world user queries.
Overall, the paper presents a thorough recipe for long context alignment and provides effective training methods to speed up the training process. However, a more in-depth analysis of the impact on general capabilities and a comparison with other methods would make the paper more comprehensive.

⚙ **LongAlign-13B-64k** (Llama-2-13B-64k trained on *LongAlign-10k* using sorted batching):
Strengths:
1. The paper presents a comprehensive solution for aligning large language models to handle long context interactions, including constructing a diverse long instruction dataset, training methods, and evaluation benchmark.
2. The authors conduct extensive experiments to demonstrate the effectiveness of their approach, showing improvements in downstream tasks without performance degradation on general benchmarks.
3. The paper provides valuable insights into the challenges and solutions for long context alignment, which is an important but understudied area.
4. The proposed LongBench-Chat benchmark is a valuable resource for evaluating long context understanding and instruction-following abilities of large language models.
5. The paper is well-written and easy to follow, with clear explanations and visuals.
Weaknesses:
1. The paper lacks a clear theoretical framework or motivation for why long context alignment is important and how it differs from short context alignment.

2. The experiments could benefit from more in-depth analysis and discussion of the results, especially in relation to the proposed solutions.
3. The evaluation on LongBench-Chat is based on a small number of human-annotated examples, which may not fully capture the diversity of real-world queries.
4. The paper does not discuss potential limitations or scalability issues with the proposed approach as the model size and context length increase.
5. The proposed loss weighting strategy is not fully explained and justified, and its impact on training stability and convergence could be further investigated.

The authors believe LongAlign-13B-64k generates a relatively good review, with only a few flaws. This case also shows the O.O.D. generalization ability of the model — it can generate reasonable review for a research paper without encountering such type of data during training.

👤 **User**: Use rap to tell about this paper.

⚙ **LongAlign-6B-64k**:
This paper presents a recipe for long context alignment, LongAlign, which addresses the challenges of training large language models (LLMs) to handle longer inputs. The key ideas of LongAlign include:
1. Data: Collecting diverse long sequences from multiple sources and using Self-Instruct to generate 10k instruction data of 8k-64k length.
2. Training method: Considering a more efficient packing strategy that packs sequences together up to the maximum length before dispatching them to GPUs for optimization. This is balanced with a loss weighting strategy to prevent sequences in packs with fewer sequences from disproportionately affecting the loss. Also proposing a sorted batching strategy that groups sequences of similar lengths to reduce intra-batch idle time.
3. Evaluation: Introducing LongBench-Chat, a series of open-ended real-world user questions of 10k-100k length, covering aspects of instruction-following like reasoning, coding, and summarization, as well as being multilingual. GPT-4 is employed to score the model's responses.
Experiments demonstrate that LongAlign effectively aligns LLMs to handle long contexts of up to 64k length without performance degradation on general benchmarks, showing the impact of data quantity, diversity, training methods, and evaluation.

⚙ **LongAlign-7B-64k**:
This paper presents LongAlign, a recipe for aligning large language models to handle long contexts. The key challenges in long context scaling include lack of a comprehensive instruction-following dataset, uneven batching in multi-GPU training, and absence of a reliable benchmark. LongAlign addresses these challenges by collecting long instruction data from 9 sources, using Self-Instruct to generate 10k instruction data of 8k-64k length, and constructing LongBench-Chat, a series of open-ended real-world user questions of 10k-100k length. The paper also proposes packing and sorted batching training methods, loss weighting for packing training, and a benchmark called LongBench-Chat to evaluate the aligned models. Experiments show that LongAlign effectively aligns models to handle contexts of up to 64k without performance degradation on general benchmarks.

⚙ **LongAlign-13B-64k**:
Yo, listen up, I got a story to spit
About a paper that'll make your head spin and flip
It's called "LongAlign: A Recipe for Long Context Alignment"
And it's all about enhancing large language models' capabilities

They tackle the challenge of scaling up models to handle long contexts

By collecting and constructing a diverse dataset of long instruction data
Training methods like packing and sorted batching are explored
To optimize efficiency and performance without compromising general capabilities

Their evaluation benchmark, LongBench-Chat, is a series of real-world user queries
To assess models' ability to follow long context instructions and provide relevant answers
They find that incorporating long instruction data during training is crucial And that the proposed methods can speed up training while improving performance

So if you're into language models and their capabilities
This paper's a must-read, it'll make you think twice
LongAlign, the recipe for long context alignment
Is the key to unlocking the full potential of large language models, that's for sure, it's a go!

Although LongAlign-6B-64k and LongAlign-7B-64k make good summaries for our paper, only LongAlign-13B-64k follows the instruction and summarizes our paper in rap. Its rap is in the right form, yet it does not rhyme.