How to Train Long-Context Language Models (Effectively)
如何有效训练长上下文语言模型

Tianyu Gao*, Alexander Wettig*, Howard Yen, Danqi Chen
Princeton Language and Intelligence, Princeton University
{tianyug, awettig, hyen, danqic}@cs.princeton.edu

Abstract

We study continued training and supervised fine-tuning (SFT) of a language model (LM) to make effective use of long-context information. We first establish a reliable evaluation protocol to guide model development-instead of perplexity or simple needle-in-a-haystack (NIAH) tests, we use a broad set of long-context tasks, and we evaluate models after SFT with instruction data as this better reveals long-context abilities. Supported by our robust evaluations, we run thorough experiments to decide the data mix for continued pre-training, the instruction tuning dataset, and many other design choices. We find that (1) code repositories and books are excellent sources of long data, but it is crucial to combine them with high-quality short data; (2) training with a sequence length beyond the evaluation length boosts long-context performance; (3) for SFT, using only short instruction datasets yields strong performance on long-context tasks. Our final model, ProLong-8B, which is initialized from Llama-3 and trained on 40B tokens, demonstrates state-of-the-art long-context performance among similarly sized models at a length of 128K. ProLong outperforms Llama-3.1- 8B-Instruct on the majority of long-context tasks despite having seen only 5% as many tokens during long-context training. Additionally, ProLong can effectively process up to 512 K tokens, one of the longest context windows of publicly available LMs. [1]

我们研究了语言模型 (LM) 的持续训练和监督微调 (SFT)，以有效利用长上下文信息。我们首先建立了一个可靠的评估协议来指导模型开发——我们使用一系列广泛的长上下文任务，而不是困惑度或简单的针在干草堆 (NIAH) 测试，并在使用指令数据进行 SFT 后评估模型，因为这更能揭示长上下文能力。在我们强有力的评估支持下，我们进行了全面的实验，以决定持续预训练的数据组合、指令调优数据集以及许多其他设计选择。我们发现:(1) 代码库和书籍是长数据的优秀来源，但将它们与高质量的短数据结合起来至关重要；(2) 使用超出评估长度的序列长度进行训练可以提升长上下文性能；(3) 对于 SFT，仅使用短指令数据集在长上下文任务上表现出色。我们的最终模型 ProLong-8B，从 Llama-3 初始化并在 40B token 上训练，在相似大小的模型中展示了 128K 长度的最先进的长上下文性能。尽管在长上下文训练期间仅看到 5% 的 token，ProLong 在大多数长上下文任务上超越了 Llama-3.1-8B-Instruct。此外，ProLong 可以有效处理多达 512 K token，这是公开可用的语言模型中最长的上下文窗口之一。[1]

Takeaways for continued training of long-context models
关于长期上下文模型持续训练的要点

- Evaluation (§2): We target a range of long-context downstream tasks instead of perplexity or needle-in-a-haystack, while checking if the short-context performance is preserved. We evaluate models after SFT, which produces a clearer signal on long-context tasks.
- Data engineering (§3): We conduct a series of ablations at a 5B-token scale. We find that using code repositories and long books as long-context data and mixing them with high-quality short-context data is crucial for both long-context performance and retaining the short-context capabilities of the pre-trained model.

- 评估 (§2): 我们针对一系列长期上下文下游任务，而不是困惑度或大海捞针，同时检查短期上下文性能是否得以保留。我们在 SFT 之后评估模型，这为长期上下文任务提供了更清晰的信号。
- 数据工程 (§3): 我们在 5B token 规模上进行了一系列消融实验。我们发现，使用代码库和长书籍作为长期上下文数据，并将其与高质量的短期上下文数据混合，对于长期上下文性能和保留预训练模型的短期上下文能力至关重要。

- Scaling the data and the length (§4): We scale up the training to 20 B tokens at a 64K training length and 20B tokens at a 512K training length. Surprisingly, training on contexts longer than the evaluation length yields additional benefits.

- 数据和长度的扩展 (§4): 我们将训练规模扩大到 20 B token，训练长度为 64K，512K 时为 20B token。令人惊讶的是，在评估长度以上的上下文进行训练会带来额外的好处。

- Supervised fine-tuning (§5): We find that SFT with standard, short-context instruction datasets is sufficient for achieving good performance. Contrary to previous study, long synthetic instruction data does not boost the result in our setting.

- 监督微调 (§5): 我们发现，使用标准的短期上下文指令数据集进行 SFT 足以实现良好的性能。与之前的研究相反，长期合成指令数据在我们的设置中并未提升结果。

- ProLong models (§6): We present our final recipe and evaluation results here. All our code, data, and models are made publically available.

- ProLong 模型 (§6): 我们在此展示我们的最终方案和评估结果。我们所有的代码、数据和模型均已公开。

1 Introduction

The ability of language models (LMs) to process extremely long inputs (for example, 128K tokens) has enabled new applications, such as book summarization or learning new tasks on the fly from many examples. However, adapting LMs to process long contexts is challenging from an infrastructure and data perspective, and many design decisions are not well understood by open-source practitioners.

语言模型 (LM) 处理极长输入 (例如，128K token) 的能力使得新的应用成为可能，例如书籍摘要或从许多示例中即时学习新任务。然而，从基础设施和数据的角度来看，调整 LM 以处理长上下文是具有挑战性的，许多设计决策并未被开源从业者很好地理解。

While many works have focused on extending the context length of pre-trained LMs with minimal training (Chen et al., 2023; Peng et al., 2024), Fu et al. (2024) find that the above methods cannot even perform the simple needle-in-a-haystack (NIAH; Kamradt, 2024) task and it is necessary to continually train the LM on long documents for billions of tokens. Frontier open-source models, such as Llama-3.1 (Dubey et al., 2024) and Jamba (Lenz et al., 2024), also employ a long-context continued training stage, followed by supervised fine-tuning (SFT) on instruction data. We adopt the same setting and study continued training and SFT of a pre-trained LM for effective long-context use.

虽然许多研究集中在以最小训练扩展预训练 LM 的上下文长度上 (Chen et al., 2023; Peng et al., 2024)，但 Fu et al. (2024) 发现上述方法甚至无法执行简单的针在干草堆中 (NIAH; Kamradt, 2024) 任务，因此有必要在长文档上对 LM 进行数十亿 token 的持续训练。前沿开源模型，如 Llama-3.1 (Dubey et al., 2024) 和 Jamba (Lenz et al., 2024)，也采用了长上下文的持续训练阶段，随后在指令数据上进行监督微调 (SFT)。我们采用相同的设置，研究预训练 LM 的持续训练和 SFT 以有效使用长上下文。

*Equal contribution and corresponding authors. [1] Our code, data, and models are available at https://github.com/princeton-nlp/ProLong.

We first establish a reliable evaluation protocol to provide a meaningful signal for model development. Most existing works rely on either perplexity or NIAH for ablating training recipes. We demonstrate that neither is robust for guiding the development and opt for a broad range of downstream applications, such as retrieval-augmented generation (RAG), long-document summarization, and many-shot in-context learning (ICL). Importantly, we also conduct our evaluations after performing SFT, even for all our ablation runs on continued pre-training. We observe that, on some long-context tasks, performance gains only emerge after SFT, which means that best design choices can differ before and after SFT.

我们首先建立一个可靠的评估协议，以提供对模型开发有意义的信号。大多数现有工作依赖于困惑度或 NIAH 来消融训练配方。我们证明这两者都不够稳健，无法指导开发，因此选择广泛的下游应用，如检索增强生成 (RAG)、长文档摘要和多示例上下文学习 (ICL)。重要的是，我们还在执行 SFT 后进行评估，即使对于我们在持续预训练上的所有消融运行。我们观察到，在某些长上下文任务中，性能提升仅在 SFT 之后出现，这意味着最佳设计选择在 SFT 之前和之后可能有所不同。

Guided by our evaluation protocol, we run comprehensive experiments with Llama-3- 8B (8K original context window; Dubey et al., 2024) to study each component of long-context continued training, including data mixture, data and length scaling, supervised fine-tuning, and many other design choices such as cross-document attention masking and position extrapolation. Many of our findings are surprising or contradictory to existing claims, for example, (1) training only on long data hurts long-context performance, (2) training on longer sequences than the evaluation length helps, and (3) SFT on only short instruction data is sufficient for good long-context performance. We outline our main takeaways and the structure of the paper in the takeaway box at the beginning of this section.

在我们的评估协议的指导下，我们对 Llama-3-8B(8K 原始上下文窗口；Dubey 等，2024) 进行了全面实验，以研究长上下文持续训练的每个组件，包括数据混合、数据和长度缩放、监督微调以及许多其他设计选择，如跨文档注意力掩蔽和位置外推。我们的许多发现令人惊讶或与现有主张相矛盾，例如，(1) 仅在长数据上训练会损害长上下文性能，(2) 在比评估长度更长的序列上训练是有帮助的，以及 (3) 仅在短指令数据上进行 SFT 就足以获得良好的长上下文性能。我们在本节开头的要点框中概述了我们的主要收获和论文结构。

Our final model, ProLong, achieves the best performance at a 128K context length among 10B-parameter models, while taking only 5% of the data budget compared to Llama-3.1's long-context training (Dubey et al., 2024). ProLong has a maximum context length of 512K tokens, making it one of the longest-context LMs available. [2]

我们的最终模型 ProLong 在 10B 参数模型中以 128K 上下文长度达到了最佳性能，同时相比于 Llama-3.1 的长上下文训练仅占用了 5% 的数据预算 (Dubey 等，2024)。ProLong 的最大上下文长度为 512K token，使其成为可用的最长上下文语言模型之一。[2]

2 Guide Model Development With Meaningful Evaluations 用有意义的评估指导模型开发

A pre-requisite for training a strong LM is having a robust evaluation suite that can guide model development while tracking its utility in real-world applications. While synthetic benchmarks like needle-in-a-haystack (NIAH; Kamradt, 2024) and RULER (Hsieh et al., 2024) have gained much popularity due to their simplicity and controllability, we are interested in a wider range of tasks that reflect practical usage, such as the ability to reason over the whole document. In the following, we describe our evaluation protocols and showcase why they are critical to our model development.

训练强大的语言模型的前提是拥有一个强大的评估套件，该套件可以在跟踪其在实际应用中的效用的同时指导模型开发。虽然像 needle-in-a-haystack(NIAH；Kamradt，2024) 和 RULER(Hsieh 等，2024) 这样的合成基准因其简单性和可控性而获得了广泛的关注，但我们对更广泛的任务感兴趣，这些任务反映了实际使用情况，例如能够对整个文档进行推理。接下来，我们将描述我们的评估协议，并展示它们为何对我们的模型开发至关重要。

2.1 Evaluate on diverse and realistic tasks **在多样化和现实的任务上进行评估**

We first make the decision to use HELMET (Yen et al., 2024b) as our evaluation suite, as it is one of the most comprehensive long-context benchmarks, covering the following tasks:

我们首先决定使用 HELMET(Yen 等，2024b) 作为我们的评估套件，因为它是最全面的长上下文基准之一，涵盖以下任务:

- Recall: Given a JSON file with random key-values pairs, retrieve the value for a key.

- 回忆: 给定一个包含随机键值对的 JSON 文件，检索某个键的值。

- RAG: Answer a question given retrieved Wikipedia documents (NQ, HotPotQA, PopQA).

- RAG: 根据检索到的维基百科文档回答问题 (NQ, HotPotQA, PopQA)。

- Re-ranking: Produce top-10 rankings from a shuffled list of documents (MSMARCO).

- 重新排序: 从打乱的文档列表中生成前 10 名排名 (MSMARCO)。

- ICL: Learn classification tasks from many in-context examples, where the #classes ranges from 6 to 151; average of 5 datasets (TREC coarse/fine, NLÜ, Banking77, Clinc-150).

- ICL: 从许多上下文示例中学习分类任务，其中类别数量范围从 6 到 151；平均 5 个数据集 (TREC 粗/细，NLÜ，Banking77，Clinc-150)。

- QA: Answer a question given a full-length book (NarrativeQA).

- QA: 根据一本完整的书籍回答问题 (NarrativeQA)。

- Summarization: Summarize long legal documents (Multi-LexSum).

- 摘要: 总结长篇法律文件 (Multi-LexSum)。

Overall, these diverse tasks reflect a range of long-context abilities including recall, reasoning, learning from context, and robustness to noisy inputs. Yen et al. (2024b) also show that HELMET produces model performance trends that are more consistent with human perceptions unlike other long-context benchmarks.

总体而言，这些多样化的任务反映了一系列长上下文能力，包括回忆、推理、从上下文中学习以及对噪声输入的鲁棒性。Yen 等人 (2024b) 还表明，HELMET 产生的模型性能趋势与人类感知更为一致，而不同于其他长上下文基准。

[2] Throughout the paper, we use binary prefixes $\mathrm{K} = 2^{10}, \mathrm{M} = 2^{20}$, and $\mathrm{B} = 2^{30}$.
[2] 在整篇论文中，我们使用二进制前缀 $\mathrm{K} = 2^{10}, \mathrm{M} = 2^{20}$ 和 $\mathrm{B} = 2^{30}$ 。

We showcase the importance of a robust evaluation suite in Table 1. As a predecessor of our work, Fu et al. (2024) only consider needle-in-a-haystack (NIAH) and perplexity during model development; evaluations on 3 tasks from HELMET reveal major short-comings of their models despite perfect NIAH scores. We also see how NIAH and even the HELMET recall task become saturated for strong models (Llama-3.1-8B vs. 70B) while other task categories continue to detect differences in their long-context abilities.

我们在表 1 中展示了强大评估套件的重要性。作为我们工作的前身，Fu 等人 (2024) 在模型开发中仅考虑了针在干草堆中 (NIAH) 和困惑度；对 HELMET 的 3 个任务的评估揭示了他们模型的主要缺陷，尽管 NIAH 得分完美。我们还看到，对于强模型 (Llama-3.1-8B 与 70B)，NIAH 甚至 HELMET 回忆任务变得饱和，而其他任务类别仍然能够检测到它们在长上下文能力上的差异。

Table 1: HELMET offers a more holistic long-context evaluation. We reproduce ? on Llama-3-8B with SFT. We report the instruct Llama versions.
HELMET 提供了更全面的长上下文评估。我们在 Llama-3-8B 上重现了 Fu 等人 (2024) 的工作，使用 SFT。我们报告了指令 Llama 版本。

		HELMET		
Models	NIAH	Recall	RAG	Re-rank
?	100	95.8	52.1	23.1
Llama-3.1-8B	100	99.4	56.3	37.0
Llama-3.1-70B	100	100	62.1	49.2

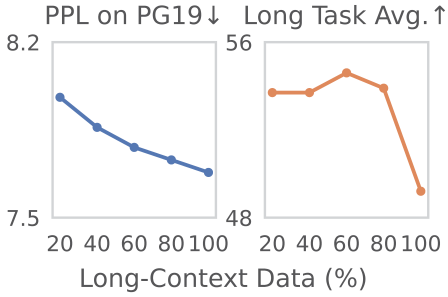

Figure 1: Making design decisions based on perplexity (PPL) is not optimal for long-context downstream tasks.
基于困惑度 (PPL) 做出设计决策对于长上下游任务并不是最佳选择。

We offer more details about the HELMET evaluation, including its careful choice of metrics, in §A.1. We did not use all tasks from HELMET for efficiency reasons and we also do not want to overfit to HELMET. If not otherwise specified, we average the performance for each category over all datasets and over evaluation lengths of 32K and 64K; for the final long-context score, we macro-average all categories.

我们在 §A.1 中提供了关于 HELMET 评估的更多细节，包括其对指标的仔细选择。出于效率原因，我们没有使用 HELMET 的所有任务，并且我们也不想对 HELMET 进行过拟合。如果没有其他说明，我们对每个类别在所有数据集和 32K 和 64K 的评估长度上进行性能平均；对于最终的长上下文得分，我们对所有类别进行宏平均。

Why not perplexity? Besides synthetic recall tasks, many previous works rely on perplexity (PPL) for evaluating long-context extensions of LMs (Chen et al., 2023; Fu et al., 2024; Lu et al., 2024), which is commonly measured on the PG19 books dataset (Rae et al., 2020). We use the ablation experiment from §3.2 to showcase why perplexity is not an indicative metric for developing long-context models. The experiment studies how the ratio of long documents affects the performance. We report both our evaluation and the perplexity measured on the last 32 K tokens of 64K-length documents from PG19. As shown in Figure 1, while using more long data continues to improve PPL, it is clear that using 100% long data significantly hurts downstream long-context performance.

为什么不使用困惑度？除了合成召回任务，许多之前的工作依赖于困惑度 (PPL) 来评估语言模型的长上下文扩展 (Chen et al., 2023; Fu et al., 2024; Lu et al., 2024)，通常在 PG19 图书数据集上进行测量 (Rae et al., 2020)。我们使用 §3.2 中的消融实验来展示为什么困惑度不是开发长上下文模型的指示性指标。该实验研究了长文档的比例如何影响性能。我们报告了我们的评估以及在 PG19 的 64K 长文档的最后 32 K 个标记上测量的困惑度。如图 1 所示，尽管使用更多的长数据继续改善 PPL，但显然使用 100% 的长数据会显著损害下游长上下文性能。

2.2 Evaluate after supervised fine-tuning **在监督微调后进行评估**

Supervised fine-tuning (SFT; Ouyang et al., 2022) is an additional training stage that fine-tunes the model on a small amount of natural-language instructions and corresponding responses; it enables a base

LM to address user queries in a chat format and has become a standard step for producing frontier LMs. Here, we consider the difference between evaluating a model before or after SFT.

监督微调 (SFT；Ouyang et al., 2022) 是一个额外的训练阶段，它在少量自然语言指令及其对应响应上微调模型；它使基础语言模型能够以聊天格式处理用户查询，并已成为生成前沿语言模型的标准步骤。在这里，我们考虑在 SFT 之前或之后评估模型的差异。

In preliminary experiments, we continue training Llama-3-8B-Base on 5B-token subsets from the data mix by Fu et al. (2024). The mix is based on SlimPajama (Soboleva et al., 2023) and upsamples long documents to constitute roughly 70% of tokens, while retaining the original domain proportions. Then we conduct SFT on several intermediate checkpoints with UltraChat (Ding et al., 2023).

在初步实验中，我们继续在 Fu et al. (2024) 的数据混合的 5B-token 子集上训练 Llama-3-8B-Base。该混合基于 SlimPajama (Soboleva et al., 2023)，并对长文档进行上采样，使其大约占标记的 70%，同时保留原始领域比例。然后，我们在几个中间检查点上使用 UltraChat (Ding et al., 2023) 进行 SFT。

We show the benchmarking results before and after SFT in Figure 2. Long-context evaluation shows clearer signals when it is conducted after SFT: (1) SFT shows that the model continues to improve with more training tokens on RAG and re-ranking, while the improvement is less clear or does not exist when evaluated before SFT. (2) SFT enables evaluation on realistic applications like QA and summarization, which require instruction following and have low performance before SFT. We also note that the variance from two random training runs is not substantially higher after the additional SFT phase. Therefore, unless otherwise specified, we report the long-context performance after SFT.

我们在Figure 2中展示了 SFT 之前和之后的基准测试结果。长上下文评估在 SFT 之后进行时显示出更清晰的信号:(1) SFT 显示模型在 RAG 和重新排序上随着训练标记的增加而持续改进，而在 SFT 之前评估时，改进不太明显或不存在。(2) SFT 使得在现实应用 (如 QA 和摘要) 上的评估成为可能，这些应用需要遵循指令，并且在 SFT 之前表现较差。我们还注意到，经过额外的 SFT 阶段后，两次随机训练运行的方差并没有显著增加。因此，除非另有说明，我们报告 SFT 之后的长上下文性能。

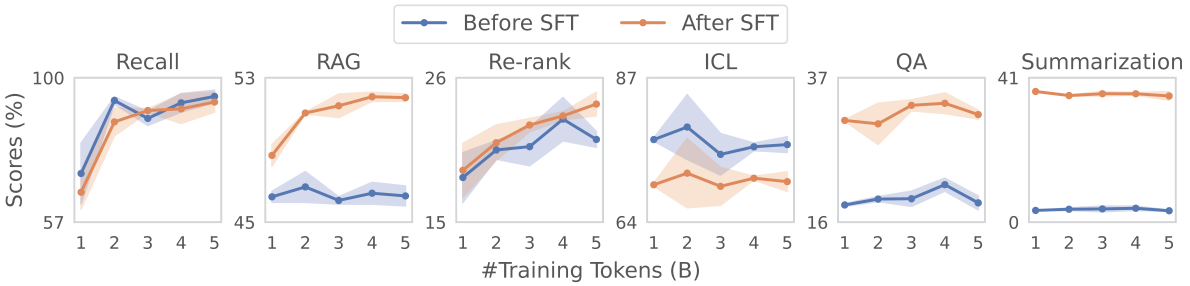

Figure 2: Improvements on RAG and re-ranking tasks are only observed when evaluating models after a (SFT) phase on instruction data. The models are trained on the pre-training data mix by ?. We report the mean and standard deviations over two training runs.
图 2: 仅在对指令数据进行监督微调 (SFT) 阶段后评估模型时，才能观察到 RAG 和重新排序任务的改进。模型是在 Fu 等人 (2024) 的预训练数据混合上训练的。我们报告了两次训练运行的均值和标准差。

We dive deeper into supervised fine-tuning in §5 and explore different training datasets, as well as the use of synthetic long instruction data. However, we find that simply fine-tuning on UltraChat remains a surprisingly competitive choice.

我们在 §5 中深入探讨监督微调，并探索不同的训练数据集，以及使用合成长指令数据。然而，我们发现仅在 UltraChat 上进行微调仍然是一个令人惊讶的竞争选择。

2.3 Check that short-context performance is preserved **检查短上下文性能是否得以保留**

Long-context abilities should not come at the expense of short-context performance, particularly since short-context evaluations cover a wider range of capabilities, e.g., world knowledge, commonsense, and

mathematical reasoning. However, short-context evaluation has largely been neglected by previous long-context research. We report on 5 tasks from the the Open LLM Leaderboard (Beeching et al., 2023): HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), ARC-challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2021), and GSM8K (Cobbe et al., 2021). We evaluate short-context performance before SFT, since this allows for a direct comparison to the base model which was used as initialization for the long-context training.

长上下文能力不应以牺牲短上下文性能为代价，特别是因为短上下文评估涵盖了更广泛的能力，例如世界知识、常识和数学推理。然而，之前的长上下文研究在很大程度上忽视了短上下文评估。我们报告了来自 Open LLM Leaderboard (Beeching 等, 2023) 的 5 项任务:HellaSwag (Zellers 等, 2019)、MMLU (Hendrycks 等, 2021)、ARC-challenge (Clark 等, 2018)、WinoGrande (Sakaguchi 等, 2021) 和 GSM8K (Cobbe 等, 2021)。我们在 SFT 之前评估短上下文性能，因为这允许与用于长上下文训练的基础模型进行直接比较。

Previous techniques deteriorate short-context performance. We show in Table 2 that both training-free position extrapolation, as well as fine-tuning with an existing long data mixture (Fu et al., 2024) do not preserve the strong performance of Llama-3-8B on standard short-context tasks. This motivates us to find data sources which retain the initial model's strong short-context performance.

之前的技术会降低短上下文性能。我们在表 2 中显示，训练无关的位置外推以及使用现有长数据混合进行微调 (Fu 等, 2024) 并未保留 Llama-3-8B 在标准短上下文任务上的强性能。这促使我们寻找能够保留初始模型强短上下文性能的数据来源。

Table 2: Applying position extrapolation (PE) to Llama-3-8B by changing the RoPE frequency base (??) or fine-tuning it on a long-context SlimPajama mixture (??) deteriorates the performance of this top-shelf pre-trained LM on short-context tasks.
通过改变 RoPE 频率基数 (§B.1) 或在长上下文 SlimPajama 混合数据集上进行微调，将位置外推 (PE) 应用于 Llama-3-8B 会降低该顶级预训练语言模型在短上下文任务上的性能。

	HSwag	MMLU	ARC-c	WG	GSM8K
Llama-3-8B	82.1	66.5	59.4	77.1	44.7
+ PE	81.5	64.7	58.1	75.5	40.1
+ SlimPajama	81.0	63.1	57.8	75.1	40.6

3 Long-Context Data Curation 长文本数据整理

The quality and composition of training data has been found to be the most important factor for LM pre-training (Penedo et al., 2023; Wettig et al., 2024; Li et al., 2024a) and is therefore a primary focus of our study. To make data decisions, we perform ablation experiments: we continue to train Llama-3-8B-Base for 5B tokens with a maximum length of 64K tokens and evaluate according to §2. See §A.4 for more details of our ablation setting.

训练数据的质量和组成被发现是语言模型预训练中最重要的因素 (Penedo et al., 2023; Wettig et al., 2024; Li et al., 2024a)，因此是我们研究的主要焦点。为了做出数据决策，我们进行消融实验:我们继续训练 Llama-3-8B-Base 5B token，最大长度为 64K token，并根据 §2 进行评估。有关我们消融设置的更多细节，请参见 §A.4。

We aim to boost the long-context task performance while preserving the short-context performance of the original model. Starting from the intuition that the data should be a mixture of long and short documents, we study these choices separately. In our ablations, the long data is comprised of single-document chunks of 64 K tokens, whereas for the short data, we construct batches by packing documents until we reach 64 K tokens per sequence.

我们的目标是在保持原始模型短文本性能的同时提升长文本任务性能。基于数据应为长短文档混合的直觉，我们分别研究这些选择。在我们的消融实验中，长数据由单文档块组成，长度为 64 K token，而短数据则通过打包文档构建批次，直到每个序列达到 64 K token。

3.1 Code repositories and books are good sources of long-context data **代码库和书籍是长文本数据的良好来源**

SlimPajama. We analyze the quantity of long data in SlimPajama (SP; ?). Table 3 shows that books account for the majority of long-context tokens. When inspecting the long data in CommonCrawl (CC), we observe that though varied in quality, it also contains some book-like content, which future work could identify via data selection methods.

SlimPajama。我们分析了 SlimPajama(SP; Soboleva et al., 2023) 中长数据的数量。表 3 显示书籍占长文本 token 的主要部分。当检查 CommonCrawl(CC) 中的长数据时，我们观察到尽管质量各异，但它也包含一些类似书籍的内容，未来的工作可以通过数据选择方法进行识别。

Table 3: Long text documents (≥64K tokens) by data sources.
按数据来源划分的长文本文档（64K token）。

Data	#Long tokens
Code Repos	98.8B
SP/Books	33.2B
SP/CC	15.3B
SP/Arxiv	5.2B
SP/GitHub	2.8B
SP/Wiki	0.1B
SP/StackEx	<0.1B
SP/C4	<0.1B

Code repositories. While only few files from GitHub reach a very long length (which also tend to be lower quality as suggested by ?), we construct an abundant source of long-context data from the Stack (?) by concatenating all files from a repository to form a single document. Unlike ?, we do not order the files based on dependencies, which should increase the distance between dependent files and reduce recency bias.

代码库。虽然来自 GitHub 的文件很少达到非常长的长度（正如 Singh 等人 (2024) 所建议的那样，这些文件的质量往往较低），我们通过将一个代码库中的所有文件连接起来形成一个单一文档，从 Stack(Kocetkov 等人，2023) 构建了丰富的长上下文数据源。与 Guo 等人 (2024) 不同，我们没有根据依赖关系对文件进行排序，这应该增加依赖文件之间的距离并减少近期偏差。

Data mixture. We train models with 60% of long-context data and 40% of our ShortMix (§3.3). Table 4 shows that using code repositories alone performs the best on stress-test recall tasks. Meanwhile, books are more broadly beneficial for in-context learning, summarization and re-ranking. An equal mix of books and code repositories achieves the best overall performance. Note that short-context task performance remains consistent due to our high-quality short data mix.

数据混合。我们使用 60% 的长上下文数据和 40% 的短数据混合 (§3.3) 来训练模型。表 4 显示，仅使用代码库在压力测试召回任务中表现最佳。同时，书籍在上下文学习、摘要和重新排序方面更具广泛的益处。书籍和代码库的均匀混合实现了最佳的整体性能。请注意，由于我们高质量的短数据混合，短上下文任务的性能保持一致。

3.2 Training only on long data hurts long-context performance **仅在长数据上训练会影响长上下文性能**

The ratio between short/long data is another crucial factor for downstream performance. Prior work either trains only on long data (Peng et al., 2024) or adds some short training data (Yen et al., 2024a; Fu et al., 2024). However, we are the first to systematically study the impact of short/long ratio.

短数据与长数据之间的比例是下游性能的另一个关键因素。之前的研究要么仅在长数据上训练 (Peng et al., 2024)，要么添加一些短训练数据 (Yen et al., 2024a; Fu et al., 2024)。然而，我们是第一个系统研究短/长比例影响的研究。

Figure 3 shows that short task performance monotonically decreases as the long data increases. The trends for long-context vary by tasks and are further complicated by SFT: On tasks like recall and RAG, the performance before SFT prefers high proportions of long data, while the performance after SFT drastically deteriorates with more long data. We hypothesize that specializing the model only on long data makes it a poor initialization for generic SFT-highlighting the importance of evaluating checkpoints after SFT (§2.2). While some long-context tasks benefit from more long data consistently (ICL) or show

no clear pattern (re-ranking), the best average performance is achieved at 60% long data and 40% short data, which we adopt for our final ProLong model.

图 3 显示，随着长数据的增加，短任务性能单调下降。长上下文的趋势因任务而异，并且由于 SFT 而进一步复杂化: 在召回和 RAG 等任务中，SFT 之前的性能更倾向于高比例的长数据，而 SFT 之后的性能在更多长数据的情况下急剧下降。我们假设仅在长数据上专门化模型使其成为通用 SFT 的较差初始化，这突显了在 SFT 之后评估检查点的重要性 (§2.2)。虽然一些长上下文任务在更多长数据的情况下持续受益 (ICL) 或没有明显模式 (重新排序)，但最佳平均性能是在 60% 长数据和 40% 短数据时实现的，这也是我们最终 ProLong 模型所采用的。

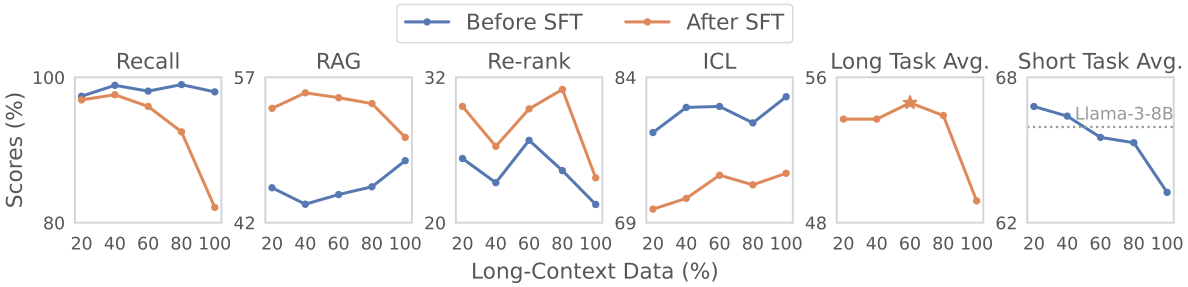

Figure 3: Impact of short/long data ratio. All models are trained on books/repos long data and our ShortMix for 5B tokens. More long data initially improves long-context performance, but then becomes impairing. More long data also consistently degrades the short-context performance.
短/长数据比例的影响。所有模型均在书籍/仓库长数据和我们的 ShortMix 上训练 5B token。更多长数据最初改善了长上下文性能，但随后变得有害。更多长数据也持续降低短上下文性能。

3.3 Choosing a high-quality short-context mix is important **选择高质量的短上下文混合非常重要**

We saw in §2.3 that it is difficult to preserve the strong performance of Llama-3-8B on short-context tasks during long-context fine-tuning. We adopt our best long-context settings (Book/repo data and 60% long/40% short) and study the impact of different short-context training mixes. We experiment with SlimPajama (Soboleva et al., 2023), FineWeb-Edu (Penedo et al., 2024), DCLM-Baseline (Li et al., 2024a), and our own ProLong ShortMix. Our ShortMix is inspired by the "stage 2 training" in MiniCPM (Hu et al., 2024a) and Dolma-1.7 (Soldaini et al., 2024), which use more knowledge-intensive, downstream-related data at the end of pre-training. Table 5 shows the composition of our ShortMix. [3]

我们在 §2.3 中看到，在长上下文微调期间，保持 Llama-3-8B 在短上下文任务上的强性能是困难的。我们采用最佳的长上下文设置 (书籍/仓库数据和 60% 长/40% 短)，并研究不同短上下文训练混合的影响。我们实验了 SlimPajama(Soboleva et al., 2023)、FineWeb-Edu(Penedo et al., 2024)、DCLM-Baseline(Li et al., 2024a) 以及我们自己的 ProLong ShortMix。我们的 ShortMix 灵感来自 MiniCPM(Hu et al., 2024a) 和 Dolma-1.7(Soldaini et al., 2024) 中的 "阶段 2 训练"，它们在预训练结束时使用更多知识密集型、与下游相关的数据。表 5 显示了我们的 ShortMix 的组成。[3]

Table 4: Our ShortMix.
我们的 ShortMix。

Components	%
FineWeb	25
FineWeb-Edu	25
Wikipedia	10
Tulu-v2	10
StackExchange	10
ArXiv	10
OpenWebMath	10

Table 5 demonstrates that the short data component has a substantial impact on both short-context and long-context downstream performance. Our curated ShortMix outperforms other short data sources on both short and long-context tasks and our data domains are particularly important for retaining

Table 5: Impact of different short data sources. The long-context performance is the average of 6 categories at the lengths of 32K and 64K.
不同短数据源的影响。长上下文性能是 32K 长度下 6 个类别的平均值和 64 K

Short Data (40%)	Long-Context	Short-Context					
	Avg.	HellaS.	MMLU	ARC-c	WG	GSM8K	Avg.
Original model (Llama-3-8B)	-	82.1	66.5	59.4	77.1	44.7	66.0
SlimPajama	52.9	81.2	63.0	58.5	76.2	41.9	64.2
FineWeb-Edu	53.0	81.0	62.6	57.7	74.4	39.4	63.0
DCLM-Baseline	52.0	82.0	65.6	59.6	77.4	39.4	64.8
orange!10!white ProLong ShortMix	54.6	81.6	65.3	58.0	76.2	46.6	65.5

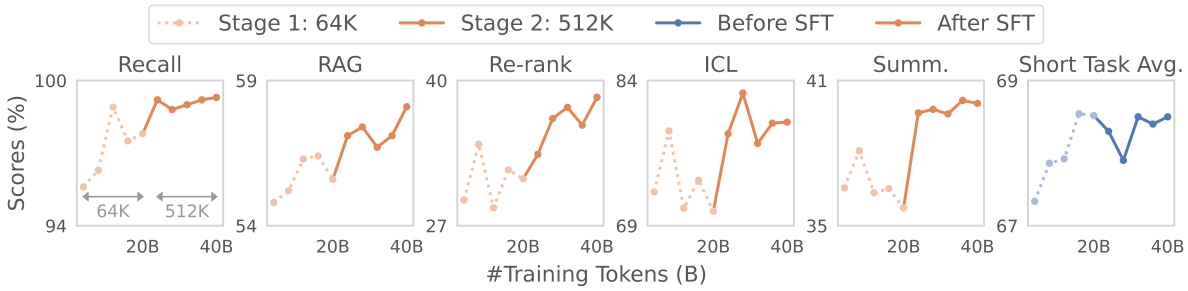

Figure 4: Performance (avg. of 32K and 64K) of our ProLong model throughout training.
我们 ProLong 模型在训练过程中的性能 (32K 和 64K 的平均值)。

Llama-3-8B's performance on mathematical reasoning. Surprisingly, we find that fine-tuning only using FineWeb-Edu—a dataset that is curated to help with knowledge-intensive tasks like MMLU—performs poorly as a short-context component, and we combine it with more diverse data sources in our ShortMix. DCLM-Baseline performs well on all short-context tasks except for GSM8K. This can likely be improved by combining with math-related datasets, but as we added the DCLM-baseline ablation at the conclusion of the project, we leave this exploration to future work.

表 6 显示短数据组件对短上下文和长上下文下游性能有显著影响。我们精心策划的 ShortMix 在短上下文和长上下文任务上均优于其他短数据源，而我们的数据领域对于保持 Llama-3-8B 在数学推理上的性能尤为重要。令人惊讶的是，我们发现仅使用 FineWeb-Edu 进行微调——这是一个旨在帮助处理知识密集型任务 (如 MMLU) 的数据集——作为短上下文组件的表现较差，因此我们将其与更多样化的数据源结合在我们的 ShortMix 中。DCLM-Baseline 在所有短上下文任务上表现良好，除了 GSM8K。通过与数学相关的数据集结合，这可能会得到改善，但由于我们在项目结束时添加了 DCLM-baseline 消融，我们将这一探索留给未来的工作。

4 Scaling the Size and Length of the Training Data **扩大训练数据的规模和长度**

Training for more steps is well-known to improve downstream tasks in regular pre-training, but little analysis has been done in the context of long-context continued training. We

众所周知，训练更多步骤可以改善常规预训练中的下游任务，但在长上下文持续训练的背景下，几乎没有进行过分析。我们

[3] Since we do not truncate documents in the short data component unnecessarily, it includes a small percentage of documents longer than 8 K . See Table 14 in the appendix for the dataset length statistics.

[3] 由于我们在短数据组件中不会不必要地截断文档，因此它包含了一小部分超过 8 K 的文档。有关数据集长度统计，请参见附录中的表 14。

incorporate the lessons from our ablation experiments and arrive at the ProLong recipe, which we describe in detail in §6. Notably, we scale up the training budget to longer sequences (up to 512K) and more tokens (20B tokens at a maximum sequence length of 64K and an additional 20B tokens at 512 K). We reset the learning rate schedule and increase the RoPE frequency base when switching from 64 K to 512 K context lengths. In this section, we analyze the impact of these decisions.

我们结合消融实验的经验教训，得出了 ProLong 配方，我们在 §6 中详细描述。值得注意的是，我们将训练预算扩大到更长的序列 (最多 512K) 和更多的标记 (在最大序列长度为 64K 时为 20B 标记，以及在 512 K 时额外的 20B 标记)。在从 64 K 切换到 512 K 上下文长度时，我们重置学习率计划并增加 RoPE 频率基。在本节中，我们分析这些决策的影响。

Increasing the number of steps helps. In Figure 4, we plot the downstream performance of intermediate checkpoints of our 40B-token runs. While the long-context performance fluctuates throughout training, we observe positive trends on recall, RAG, re-ranking, and summarization. For short-context tasks, we observe the average performance initially drops from the initialization, but gradually recovers. Performance again drops when switching from 64K to 512K sequence length, but also recovers with additional training.

增加步骤数量是有帮助的。在图 4 中，我们绘制了我们 40B-token 运行的中间检查点的下游性能。尽管长上下文性能在训练过程中波动，但我们观察到召回率、RAG、重新排序和摘要的正向趋势。对于短上下文任务，我们观察到平均性能最初从初始化时下降，但逐渐恢复。当从 64K 切换到 512K 序列长度时，性能再次下降，但在额外训练后也会恢复。

Increasing the training length beyond the evaluation length helps. One might assume that we should train long-context models on the maximum sequence length that we want the model to support. Many works even emphasize extrapolation to even longer sequences at inference time (Press et al., 2022; Xiao et al., 2024b;a; Yen et al., 2024a; Chen et al., 2023). In contrast, we observe that training on a longer sequence length (512K tokens) substantially improves the long-context performance at a shorter evaluation length (64K tokens).

将训练长度增加到超出评估长度是有帮助的。人们可能会假设我们应该在希望模型支持的最大序列长度上训练长上下文模型。许多研究甚至强调在推理时对更长序列的外推 (Press et al., 2022; Xiao et al., 2024b;a; Yen et al., 2024a; Chen et al., 2023)。相反，我们观察到在更长的序列长度 (512K tokens) 上训练显著改善了在较短评估长度 (64K tokens) 下的长上下文性能。

We establish this by initializing with a model that was trained for 20B tokens at 64 K and either (1) continuing training at 64 K , or (2) switching to the 512 K training. We use the same hyperparameters and data mixtures in either experiment. We evaluate a checkpoint after 4B training tokens at a evaluation length of 64 K . Comparing the two runs in Table 7, we see consistent gains from switching to the 512 K training length. [4]

我们通过初始化一个在 64 K 上训练了 20B tokens 的模型来建立这一点，并且 (1) 在 64 K 上继续训练，或 (2) 切换到 512 K 训练。我们在任一实验中使用相同的超参数和数据混合。在评估长度为 64 K 的情况下，我们在 4B 训练 tokens 后评估一个检查点。比较表 7 中的两个运行，我们看到切换到 512 K 训练长度的一致增益。[4]

Table 7: Impact of training models on different sequence lengths. All the results are evaluated at a sequence length of 64 K . We see that training at a maximum length beyond the evaluation context window consistently improves the long-context performance.

表 7: 在不同序列长度上训练模型的影响。所有结果均在序列长度 64 K 下评估。我们看到，在超出评估上下文窗口的最大长度上训练始终改善了长上下文性能。

Max Seq. Length	Recall	RAG	Re-rank	ICL
ProLong 64K training (20B)	96.5	52.7	22.8	70.6
+4B 64K training	95.0	56.4	28.0	78.8
+4B 512K training	98.5	56.9	32.9	79.2

最大序列长度	召回	检索增强生成	重新排序	示例学习
ProLong 64K 训练 (20B)	96.5	52.7	22.8	70.6
+4B 64K 训练	95.0	56.4	28.0	78.8
+4B 512K 训练	98.5	56.9	32.9	79.2

5 Supervised Fine-Tuning for Long-Context LMs 监督微调用于长上下文语言模型

In this section, we study how to best enable long-context language models to follow instructions. We focus on supervised fine-tuning on instruction datasets (Ouyang et al., 2022) and leave reinforcement learning and preference optimization for future work.

在本节中，我们研究如何最佳地使长上下文语言模型遵循指令。我们专注于在指令数据集上进行监督微调 (Ouyang et al., 2022)，并将强化学习和偏好优化留待未来工作。

All our experiments in this section use the ProLong base model, which was trained for 40B tokens at a maximum sequence length of 512 K . In comparison, open-source instruction data are very short, e.g., UltraChat (Ding et al., 2023) conversations have 1.2K tokens on average and 4.1K tokens maximum. To bridge this gap, several works (Xiong et al., 2023; Dubey et al., 2024; Xiong et al., 2024) have proposed to generate long instruction data synthetically.

本节中的所有实验使用 ProLong 基础模型，该模型在最大序列长度为 512 K 的情况下训练了 40B token。相比之下，开源指令数据非常短，例如，UltraChat(Ding et al., 2023) 对话的平均长度为 1.2K token，最大为 4.1K token。为了弥补这一差距，几项工作 (Xiong et al., 2023；Dubey et al., 2024；Xiong et al., 2024) 提出了合成生成长指令数据的方法。

We consider three popular SFT datasets—UltraChat (Ding et al., 2023), Tulu-v2 (Ivison et al.,2023), ShareGPT [5] - and three sources of synthetic data: For synthetic QA, we prompt Llama-3-8B-Instruct to generate a question-and-answer pair given a random chunk from a long document; we reuse the QA pairs for synthetic RAG but we present a random list of chunks from the document to mimic retrieved passages; for synthetic summarization, we generate summaries for long books via recursive summarization (Wu et al., 2021). For all synthetic data, we write several templates, which we sample at random to increase diversity. More details can be found in §A.5. We always use a combination of 40% synthetic QA, 30% synthetic RAG, and 30% synthetic summarization in our synthetic instruction dataset. The hyperparameters for the instruction tuning experiments can be found in Table 9.

我们考虑三个流行的 SFT 数据集——UltraChat(Ding et al., 2023)、Tulu-v2(Ivison et al., 2023)、ShareGPT [5] ——以及三种合成数据来源: 对于合成 QA，我们提示 Llama-3-8B-Instruct 生成一个问题和答案对，给定来自长文档的随机片段；我们重用 QA 对于合成 RAG，但我们呈现来自文档的随机片段列表，以模拟检索到的段落；对于合成摘要，我们通过递归摘要生成长书籍的摘要 (Wu et al., 2021)。对于所有合成数据，我们编写了几个模板，并随机抽样以增加多样性。更多细节可以在 §A.5 中找到。我们在合成指令数据集中始终使用 40% 的合成 QA、30% 的合成 RAG 和 30% 的合成摘要的组合。指令调优实验的超参数可以在表 9 中找到。

Short-context instruction data yields strong long-context results. We first establish that UltraChat outperforms Tulu-v2 and ShareGPT in Table 22. We therefore use it when studying the ratio of synthetic long-context instruction data in Table 8. Surprisingly, we find that adding synthetic data does not improve the performance on these very long-context tasks, and adding even as little as 1% synthetic data hurts

[4] While we demonstrate the benefit of longer data, we note that training with longer sequences is more expensive, and may therefore not be the computationally optimal choice.

[4] 虽然我们展示了更长数据的好处，但我们注意到，使用更长序列进行训练的成本更高，因此可能不是计算上最优的选择。

the performance in our setting. Therefore, we use only short-context UltraChat data for SFT of our final ProLong model.

短上下文指令数据产生强大的长上下文结果。我们首先确定 UltraChat 在表 22 中优于 Tulu-v2 和 ShareGPT。因此，在研究表 8 中合成长上下文指令数据的比例时，我们使用它。令人惊讶的是，我们发现添加合成数据并没有改善这些非常长上下文任务的性能，甚至添加 1% 的合成数据也会在我们的设置中损害性能。因此，我们仅使用短上下文的 UltraChat 数据来对我们最终的 ProLong 模型进行 SFT。

Table 8: Effect of different ratios of synthetic SFT data (mixed with UltraChat). We report the 32K-and-64K-averaged performance except tasks marked with † , which are evaluated at 512 K for stress testing. The number of percentage is based on #tokens, not #samples.

表 8: 合成 SFT 数据 (与 UltraChat 混合) 不同比例的效果。我们报告 32K 和 64K 平均性能，除了标记为 † 的任务，这些任务在 512 K 进行压力测试评估。百分比的数量是基于 #tokens，而不是 #samples。

% Synthetic Data	JsonKV †	RAG	Re-rank	ICL	$QA^{\dagger}$	Summ. †	Avg.
0%	65.7	58.1	38.5	80.3	49.7	42.1	55.7
1%	61.5	57.0	38.3	80.8	45.3	41.5	54.1
3%	62.0	56.4	37.9	80.6	44.8	39.5	53.5
10%	70.3	55.5	36.1	80.6	41.7	39.4	53.9
50%	45.8	48.8	18.8	70.5	42.3	33.3	43.3

% 合成数据	JsonKV †	检索增强生成	重新排序	即时学习	$QA^{\dagger}$	总结 †	平均值
0%	65.7	58.1	38.5	80.3	49.7	42.1	55.7
1%	61.5	57.0	38.3	80.8	45.3	41.5	54.1
3%	62.0	56.4	37.9	80.6	44.8	39.5	53.5
10%	70.3	55.5	36.1	80.6	41.7	39.4	53.9
50%	45.8	48.8	18.8	70.5	42.3	33.3	43.3

Why do our conclusions about synthetic data differ from previous work? We offer the following hypotheses: (1) Previous work like Xiong et al. (2024); Bai et al. (2024a) may have insufficient long-context training and the synthetic data acts as additional long-context training data. (2) Our instruction dataset is much smaller compared to the private instruction data used for Llama-3.1 (Dubey et al., 2024) —it is possible that when using an extensive short instruction dataset, mixing in synthetic long data avoids the model from degenerating on long-context tasks.

我们关于合成数据的结论为何与之前的工作不同？我们提出以下假设:(1) 之前的工作如 Xiong 等 (2024)；Bai 等 (2024a) 可能在长上下文训练上不足，而合成数据作为额外的长上下文训练数据。(2) 我们的指令数据集与用于 Llama-3.1 的私有指令数据相比要小得多 (Dubey 等，2024)——当使用大量短指令数据集时，混合合成的长数据可能避免模型在长上下文任务上退化。

6 The ProLong Model: Recipe and Results

6 ProLong **模型: 配方和结果**

6.1 Final recipe **最终配方**

We summarize the training recipe for ProLong in Table 9. Our final model starts from the Llama-3-8B-Instruct model and is trained on 64K sequence length for 20B tokens. It is then further trained on 512K sequence length for 20B tokens (ProLong base), which we achieve using sequence parallelism (Li et al., 2023). We obtain the final ProLong model via SFT of the base model on UltraChat. One small difference on the data mixture between our ablations and the final model is that we mix in 3% high-quality textbooks (Chevalier et al., 2024), as book-like data are shown to be beneficial for long-context

(§3.1) and textbooks are highly educational. This also slightly changes the proportions of ShortMix. You can find more details about our data processing (§A.2) and the training stack (§A.3) in the appendix. In the following, we elaborate on several carefully ablated design choices in our recipe.

我们在表 9 中总结了 ProLong 的训练配方。我们的最终模型从 Llama-3-8B-Instruct 模型开始，并在 64K 序列长度上训练 20B token。然后在 512K 序列长度上进一步训练 20B token(ProLong 基础)，我们通过序列并行性实现这一点 (Li 等，2023)。我们通过在 UltraChat 上对基础模型进行 SFT 获得最终的 ProLong 模型。我们在消融实验与最终模型之间的数据混合的一个小差异是，我们混入了 3% 的高质量教科书 (Chevalier 等，2024)，因为书籍类数据被证明对长上下文有益 (§3.1)，而教科书具有很高的教育价值。这也稍微改变了 ShortMix 的比例。有关我们的数据处理 (§A.2) 和训练堆栈 (§A.3) 的更多细节，请参见附录。接下来，我们详细阐述我们配方中几个经过仔细消融的设计选择。

Table 9: The training recipe for ProLong.

表 9:ProLong 的训练配方。

Continued Long-context Training		
Data	30% code repos, 30% books, 3% textbooks, 37% ShortMix	
	ShortMix:	27% FineWeb-Edu, 27% FineWeb, 11% Tulu-v2, 11% StackExchange, 8% Wikipedia, 8% OpenWebMath, 8% ArXiv
Length Curriculum	Stage 1 (64K):	Code repos, books, and textbooks at length 64 K
	Stage 2 (512K):	Code repos: 50% at length 512K, 50% at length 64K Books: 17% at length 512K, 83% at length 64K Textbooks at length 512K
Steps	Stage 1: 20B tokens (2.2K H100 hours), Stage 2: 20B tokens (12.2K H100 hours)	
Model	Initialization: RoPE:	Llama-3-8B-Instruct (original RoPE base freq. 5×10^5) Stage 1: 8×10^6 , Stage 2: 1.28×10^8
	Attention:	Full attention with cross-document attention masking
Optim.	LR:	lamW (weight decay $= 0.1, \beta_1 = 0.9, \beta_2 = 0.95$) $1e-5$ with 10% warmup and cosine decay to $1e-6$, each stage
	Batch size:	4M tokens for stage 1, 8M tokens for stage 2
Supervised Fine-tuning (SFT)		
Data	UltraChat	
Steps	1B tokens	
Optim.	AdamW (weight decay $= 0.1, \beta_1 = 0.9, \beta_2 = 0.95$) LR $= 2e-5$ (cosine decay to $2e-6$), warmup $= 5\%$ Batch size $= 4$M tokens	

持续的长上下文训练		
数据	30% 代码库，30% 书籍，3% 教科书，37% ShortMix	
	ShortMix:	27% FineWeb-Edu，27% FineWeb，11% Tulu-v2，11% StackExchange，8% 维基百科，8% OpenWebMath，8% ArXiv
长度课程	阶段 1 (64K):	代码库、书籍和教科书的详细信息 64 K
	阶段 2 (512K):	代码库:50% 为 512K，50% 为 64K 书籍:17% 为 512K，83% 为 64K 教科书为 512K
步骤	阶段 1:20B token (2.2K H100 小时)，阶段 2:20B token (12.2K H100 小时)	
模型	初始化:RoPE:	Llama-3-8B-Instruct(原始 RoPE 基频 5×10^5) 第一阶段: 8×10^6 ，第二阶段: 1.28×10^8
	注意力:	全注意力与跨文档注意力掩蔽
优化。	学习率:	lamW(权重衰减 $= 0.1, \beta_1 = 0.9, \beta_2 = 0.95$) $1e-5$ ，具有 10% 的预热和余弦衰减到 $1e-6$ ，每个阶段
	批量大小:	阶段 1 为 4M token，阶段 2 为 8M token
监督微调 (SFT)		
Data	UltraChat	
步骤	1B token	
优化。	AdamW(权重衰减 $= 0.1, \beta_1 = 0.9, \beta_2 = 0.95$) LR $= 2e-5$ (余弦衰减至 $2e-6$)，预热 $= 5\%$ 批量大小 $= 4$M token	

RoPE frequency base tuning. We find that changing the RoPE (Su et al., 2021) frequency base to achieve position extrapolation (Xiong et al., 2023; emozilla, 2023) significantly improves long-context performance, even with a significant amount of training. §B. 1 shows our ablation on the best RoPE base to use. While the original Llama models use a RoPE base of 10^5 , we use a base of 8×10^6 for the 64 K setting and 1.28×10^8 for the 512 K setting.

RoPE 频率基准调优。我们发现，改变 RoPE (Su et al., 2021) 频率基准以实现位置外推 (Xiong et al., 2023; emozilla, 2023) 显著提高了长上下文性能，即使经过大量训练。§B. 1 显示了我们对最佳 RoPE 基准的消融实验。虽然原始的 Llama 模型使用的 RoPE 基准为 10^5 ，我们在 64 K 设置中使用 8×10^6 作为基准，在 512 K 设置中使用 1.28×10^8 作为基准。

Disabling cross-document attention. Ding et al. (2024a) show that masking out attention across document boundaries improve model performance and this was also used during Llama-3 pre-training (Dubey et al., 2024). In §B.2, we show that disabling cross-document attention in continued training

[5] https://huggingface.co/datasets/RyokoAI/ShareGPT52K.

[5] https://huggingface.co/datasets/RyokoAI/ShareGPT52K。

benefits both the short and long-context performance. Disabling cross-document attention can also result in higher training throughput, which we describe in more detail in §A.3.

禁用跨文档注意力。Ding et al. (2024a) 表明，屏蔽跨文档边界的注意力可以提高模型性能，这在 Llama-3 预训练期间也得到了应用 (Dubey et al., 2024)。在 §B.2 中，我们展示了在继续训练中禁用跨文档注意力对短上下文和长上下文性能的益处。禁用跨文档注意力还可以导致更高的训练吞吐量，我们在 §A.3 中对此进行了更详细的描述。

Starting from Llama-3-8B-Instruct. While we conduct all our long-context training ablations with the base model of Llama-3-8B, we use Llama-3-8B-Instruct as the initialization for the final ProLong model. §B. 3 shows that while slightly improving the long-context performance, Llama-3-8B-Instruct significantly enhances the short-context performance.

从 Llama-3-8B-Instruct 开始。虽然我们对所有长上下文训练的消融实验都使用 Llama-3-8B 的基础模型，但我们使用 Llama-3-8B-Instruct 作为最终 ProLong 模型的初始化。§B. 3 显示，虽然略微提高了长上下文性能，但 Llama-3-8B-Instruct 显著增强了短上下文性能。

6.2 ProLong performance

We present the final HELMET evaluation results of ProLong in Table 10. We compare to a number of frontier long-context LMs, namely MegaBeam6, Llama-3.1 (Dubey et al., 2024), Qwen2 (Yang et al., 2024a), Phi-3 (Abdin et al., 2024), Mistral-Nemo [7] , Jamba-1.5 (Lenz et al., 2024), Claude-3.5-Sonnet (Anthropic, 2024), Gemini-1.5 (Reid et al., 2024), and GPT- 4o (Achiam et al., 2023).

我们在表 10 中展示了 ProLong 的最终 HELMET 评估结果。我们与多个前沿长上下文语言模型进行了比较，即 MegaBeam6、Llama-3.1 (Dubey et al., 2024)、Qwen2 (Yang et al., 2024a)、Phi-3 (Abdin et al., 2024)、Mistral-Nemo [7] 、Jamba-1.5 (Lenz et al., 2024)、Claude-3.5-Sonnet (Anthropic, 2024)、Gemini-1.5 (Reid et al., 2024) 和 GPT-4o (Achiam et al., 2023)。

ProLong outperforms all 10B-scale models on our long-context evaluation. Notably, ProLong outperforms Llama-3.1-8B-Instruct on all categories except summarization. ProLong

ProLong 在我们的长上下文评估中超越了所有 10B 规模的模型。值得注意的是，ProLong 在除摘要外的所有类别中都超越了 Llama-3.1-8B-Instruct。

Table 10: Our main evaluation results on HELMET (Yen et al., 2024b; details in §A.1). All results are averaged over sequence lengths of 32K, 64K, and 128K. For all models, we use the corresponding instruction version. ProLong is one of the best performing 10B-scale LMs. The complete set of results can be found in §C .

表 10: 我们在 HELMET 上的主要评估结果 (Yen 等，2024b；详细信息见 §A.1)。所有结果都是在序列长度为 32K、64K 和 128K 的基础上平均得出的。对于所有模型，我们使用相应的指令版本。ProLong 是表现最佳的 10B 规模语言模型之一。完整的结果集可以在 §C 中找到。

[6] https://huggingface.co/aws-prototyping/MegaBeam-Mistral-7B-512k.

[6] https://huggingface.co/aws-prototyping/MegaBeam-Mistral-7B-512k.

[7] https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407.

[7] https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407.

Model	Max Len.	Recall	RAG	ICL	Re-rank	QA	Summ.	Avg.
ProLong (8B)	512K	99.4	66.0	81.1	33.2	40.8	40.5	60.2
MegaBeam-Mistral (7B)	512K	99.4	58.1	82.1	22.1	33.7	43.6	56.5
Meta-Llama-3.1 (8B)	128K	98.7	62.8	79.7	26.6	40.4	46.1	59.0
Qwen2 (7B)	128K	34.4	43.4	54.8	4.6	23.3	38.5	33.2
Phi-3-small (7B)	128K	74.8	60.6	82.0	18.5	34.1	42.4	52.1
Mistral-Nemo (12B)	128K	24.9	48.1	82.0	4.7	37.7	37.0	39.1
Jamba-1.5-Mini (12B/52B)	256K	87.7	61.3	88.4	25.9	42.0	38.6	57.3
Meta-Llama-3.1 (70B)	128K	98.5	65.9	80.0	39.4	47.2	51.1	63.7
Claude-3.5-Sonnet	200K	99.4	44.0	79.3	19.9	38.1	49.2	55.0
Gemini-1.5-Pro	2M	94.2	71.4	78.9	65.3	44.4	56.2	68.4
GPT-40	128K	99.9	71.5	86.7	59.6	47.0	55.7	70.1

模型	最大长度	召回率	检索增强生成	即刻学习	重新排序	问答	总结	平均
ProLong (8B)	512K	99.4	66.0	81.1	33.2	40.8	40.5	60.2
MegaBeam-Mistral (7B)	512K	99.4	58.1	82.1	22.1	33.7	43.6	56.5
Meta-Llama-3.1 (8B)	128K	98.7	62.8	79.7	26.6	40.4	46.1	59.0
Qwen2 (7B)	128K	34.4	43.4	54.8	4.6	23.3	38.5	33.2
Phi-3-small (7B)	128K	74.8	60.6	82.0	18.5	34.1	42.4	52.1
Mistral-Nemo (12B)	128K	24.9	48.1	82.0	4.7	37.7	37.0	39.1
Jamba-1.5-Mini (12B/52B)	256K	87.7	61.3	88.4	25.9	42.0	38.6	57.3
Meta-Llama-3.1 (70B)	128K	98.5	65.9	80.0	39.4	47.2	51.1	63.7
Claude-3.5-Sonnet	200K	99.4	44.0	79.3	19.9	38.1	49.2	55.0
Gemini-1.5-Pro	2M	94.2	71.4	78.9	65.3	44.4	56.2	68.4
GPT-40	128K	99.9	71.5	86.7	59.6	47.0	55.7	70.1

Table 12: Results on the NoCha benchmark (Karpinska et al., 2024). [9] ProLong is the only model that achieves above-random performance in the < 75 K category and it consistently beats Llama-3.1. Different from the original NoCha leaderboard, we report the average accuracy over all test instances without filtering the test examples based on the model's context window lengths.

表 12: 在 NoCha 基准上的结果 (Karpinska et al., 2024)。[9] ProLong 是唯一一个在 < 75 K 类别中实现超随机性能的模型，并且它始终优于 Llama-3.1。与原始 NoCha 排行榜不同，我们报告所有测试实例的平均准确率，而不根据模型的上下文窗口长度过滤测试示例。

Model	Max Len.	<75K	75K-127K	127K-180K	>180K
ProLong (8B)	512K	28.4	17.0	13.1	20.3
MegaBeam-Mistral (7B)	512K	19.8	18.3	17.5	15.6
Meta-Llama-3.1 (8B)	128K	17.3	16.4	0.0	0.0
Mistral-Nemo (12B)	128K	13.6	0.4	0.0	0.0
Jamba-1.5-Mini (12B/52B)	256K	27.2	28.0	24.4	6.2
Meta-Llama-3.1 (70B)	128K	42.0	25.0	0.0	0.0
Gemini-1.5-Pro	2M	24.7	38.8	35.3	46.9
GPT-40	128K	55.6	58.4	0.0	0.0

模型	最大长度	<75K	75K-127K	127K-180K	>180K
ProLong (8B)	512K	28.4	17.0	13.1	20.3
MegaBeam-Mistral (7B)	512K	19.8	18.3	17.5	15.6
Meta-Llama-3.1 (8B)	128K	17.3	16.4	0.0	0.0
Mistral-Nemo (12B)	128K	13.6	0.4	0.0	0.0
Jamba-1.5-Mini (12B/52B)	256K	27.2	28.0	24.4	6.2
Meta-Llama-3.1 (70B)	128K	42.0	25.0	0.0	0.0
Gemini-1.5-Pro	2M	24.7	38.8	35.3	46.9
GPT-40	128K	55.6	58.4	0.0	0.0

achieves this with only 5% of Llama-3.1' s long-context data budget (40B vs. 800B tokens). We also showcase the strength of ProLong with several QA examples in Table 23.

仅使用 Llama-3.1 的 5% 长上下文数据预算 (40B 对比 800B token) 来实现这一点。我们还在表 23 中展示了 ProLong 在多个 QA 示例中的强大能力。

Since most existing models do not support more than 128 K tokens, to showcase ProLong's 512K context length, we stress test ProLong on the QA and summarization tasks from 32K to 512 K[8] . Table 11 shows that ProLong continues to improve at a longer context window.

由于大多数现有模型不支持超过 128 K token，为了展示 ProLong 的 512K 上下文长度，我们对 ProLong 在 32K 到 512 K[8] 的 QA 和摘要任务进行了压力测试。表 11 显示 ProLong 在更长的上下文窗口中持续改进。

Table 11: ProLong at 512K.

表 11:ProLong 在 512K。

	32K	64K	128K	512K
OA	31.7	43.7	46.7	49.7
Summ	40.4	39.8	41.5	42.1

	32K	64K	128K	512K
开放获取	31.7	43.7	46.7	49.7
摘要	40.4	39.8	41.5	42.1

Besides HELMET, we also evaluate our models on NoCha (Karpinska et al., 2024) - a claim verification dataset on 67 recently published English fictional books. We chose this dataset because (1) it minimizes the data contamination problem as all the books are unlikely to exist in the model pre-training data; (2) all the claims are written by human readers and require global reasoning. Each test instance contains two contradictory claims, and the models must correctly judge both to pass.

除了 HELMET，我们还在 NoCha (Karpinska et al., 2024) 上评估我们的模型——这是一个关于 67 本最近出版的英文虚构书籍的声明验证数据集。我们选择这个数据集是因为 (1) 它最小化了数据污染问题，因为所有书籍不太可能存在于模型的预训练数据中；(2) 所有声明都是由人类读者撰写的，并且需要全球推理。每个测试实例包含两个矛盾的声明，模型必须正确判断这两个声明才能通过。

Table 12 demonstrates the NoCha evaluation results. Among 10B-scale models, ProLong achieves the best accuracy on the extremely long test instances ($>$ 180 K) ; on test instances $<$ 75 K tokens, ProLong significantly outperforms other models and is the only model that is better than random guessing (25%). This further showcases the strength of our training recipe and the ProLong model.

表 12 展示了 NoCha 评估结果。在 10B 规模的模型中，ProLong 在极长的测试实例上取得了最佳准确率 ($>$ 180 K) ；在测试实例 $<$ 75 K 的标记上，ProLong 显著优于其他模型，并且是唯一一个表现优于随机猜测 (25%) 的模型。这进一步展示了我们的训练方案和 ProLong 模型的优势。

[8] In QA and summarization, we truncate the documents at the evaluation length; hence an effective long-context model should demonstrate better performance on longer lengths.

[8] 在问答和摘要中，我们在评估长度处截断文档；因此，一个有效的长上下文模型应该在更长的长度上表现更好。

[9] https://github.com/marzenakrp/nocha.NoCha has a private test set and all evaluation is done by the NoCha authors. Hence, we report models from Table 10 that are also on the NoCha leaderboard.

[9] https://github.com/marzenakrp/nocha.NoCha 有一个私有测试集，所有评估均由 NoCha 作者完成。因此，我们报告表 10 中的模型，这些模型也在 NoCha 排行榜上。

7 Related Work

Adapting existing LMs for long contexts. Many works explore extending the LM context windows with minimal training, either by position extrapolation (Chen et al., 2023; Peng et al., 2024; Chen et al., 2024; Ding et al., 2024b; Liu et al., 2024a; Zhang et al., 2024b; Zhu et al., 2024; Zhao et al., 2024; Wu et al., 2024; Hu et al., 2024b) or manipulating the attention patterns (Chen et al., 2024; Xiao et al., 2024b;a; Bertsch et al., 2023; Jin et al., 2024). Yoshida et al. (2020); Choromanski et al. (2021); Chevalier et al. (2023) instead explore the idea of compressing the long contexts into shorter forms. However, Fu et al. (2024); Lu et al. (2024) show that using full attention, applying simple position extrapolation, and fine-tuning the model on long documents reach much stronger results.

将现有的语言模型适应于长上下文。许多研究探索通过最小训练扩展语言模型的上下文窗口，方法包括位置外推 (Chen et al., 2023; Peng et al., 2024; Chen et al., 2024; Ding et al., 2024b; Liu et al., 2024a; Zhang et al., 2024b; Zhu et al., 2024; Zhao et al., 2024; Wu et al., 2024; Hu et al., 2024b) 或操控注意力模式 (Chen et al., 2024; Xiao et al., 2024b;a; Bertsch et al., 2023; Jin et al., 2024)。Yoshida et al. (2020); Choromanski et al. (2021); Chevalier et al. (2023) 则探索将长上下文压缩为更短形式的想法。然而，Fu et al. (2024); Lu et al. (2024) 表明，使用完整注意力、应用简单位置外推以及在长文档上微调模型可以达到更强的结果。

Llama 3.1 (Dubey et al., 2024) and Jamba (Lieber et al., 2024) achieve long-context capabilities by adding a long-context continued training stage between standard pre-training and supervised fine-tuning, which is the setting we follow. Fu et al. (2024) study the data engineering for this setting and argue that 0.5 B tokens of domain-balanced, length-upsampled data is sufficient for acquiring the long-context recall ability-which we show is not sufficient if a more holistic evaluation is taken. Xiong et al. (2023); Dubey et al. (2024); Lieber et al. (2024); Xiong et al. (2024); An et al. (2024b); Bai et al. (2024a) also adopt synthetically-generated long data in the SFT stage; however, we find that using standard, short-context instruction data achieves the best long-context results in our setting.

Llama 3.1 (Dubey et al., 2024) 和 Jamba (Lieber et al., 2024) 通过在标准预训练和监督微调之间添加一个长上下文持续训练阶段来实现长上下文能力，这是我们遵循的设置。Fu et al. (2024) 研究了这一设置的数据工程，并认为 0.5 B 领域平衡、长度上采样的数据的 tokens 足以获得长上下文回忆能力——我们表明如果进行更全面的评估，这并不足够。Xiong et al. (2023); Dubey et al. (2024); Lieber et al. (2024); Xiong et al. (2024); An et al. (2024b); Bai et al. (2024a) 也在 SFT 阶段采用了合成生成的长数据；然而，我们发现使用标准的短上下文指令数据在我们的设置中取得了最佳的长上下文结果。

Efficient long-context architectures. There have been many efforts in designing more efficient architectures, for example, linear attention/RNNs (Gu & Dao, 2023; Dao & Gu, 2024; Ma et al., 2022; Sun et al., 2023; Peng et al., 2023; Yang et al., 2024b), and alternative attention architectures (Rubin & Berant, 2023; Sun et al., 2024; Yen et al., 2024a). However, they often require training from scratch and many have the inherent limitations in terms of long-context recall (Jelassi et al., 2024; Arora et al., 2024). Recent works explore hybrid models (Waleffe et al., 2024; Lieber et al., 2024)) or distilling existing LMs into hybrid models (Wang et al., 2024) and show promising results.

高效的长上下文架构。设计更高效架构的努力已经很多，例如线性注意力/RNNs (Gu & Dao, 2023; Dao & Gu, 2024; Ma et al., 2022; Sun et al., 2023; Peng et al., 2023; Yang et al., 2024b)，以及替代注意力架构 (Rubin & Berant, 2023; Sun et al., 2024; Yen et al., 2024a)。然而，它们通常需要从头开始训练，并且在长上下文回忆方面存在固有的局限性 (Jelassi et al., 2024; Arora et al., 2024)。最近的工作探索了混合模型 (Waleffe et al., 2024; Lieber et al., 2024) 或将现有的语言模型提炼为混合模型 (Wang et al., 2024)，并显示出良好的结果。

Long-context evaluation. Many benchmarks have been proposed for long-context evaluation (Shaham et al., 2023; Hsieh et al., 2024; Krishna et al., 2023; Zhang et al., 2024a; An et al., 2024a; Bai et al., 2024b) There are works studying particular aspects of long-context LMs as well, such as positional bias (Liu et al., 2024b), in-context learning (Bertsch et al., 2024; Li et al., 2024b), and book-length summarization (Kim et al., 2024). In this work, we follow Yen et al. (2024b) for its diverse application coverage and reliable evaluations.

长上下文评估。已经提出了许多用于长上下文评估的基准 (Shaham et al., 2023; Hsieh et al., 2024; Krishna et al., 2023; Zhang et al., 2024a; An et al., 2024a; Bai et al., 2024b)。也有研究特定方面的长上下文语言模型的工作，例如位置偏差 (Liu et al., 2024b)、上下文学习 (Bertsch et al., 2024; Li et al., 2024b) 和书籍长度的摘要 (Kim et al., 2024)。在这项工作中，我们遵循 Yen et al. (2024b) 的研究，因为它具有多样的应用覆盖和可靠的评估。

8 Conclusion

We study the problem of given a short-context pre-trained LM, how to most effectively continually pre-train and SFT the model to be long-context. We conduct thorough ablations on each component and many of our findings contradict existing practices or beliefs. We use all the findings to produce ProLong, a new state-of-the-art long-context LM. We release all our code, data, and models publicly and hope that our findings will boost research and applications of long-context LMs.

我们研究了在给定短上下文预训练语言模型的情况下，如何最有效地持续预训练和微调模型以实现长上下文。我们对每个组件进行了彻底的消融实验，许多发现与现有的实践或信念相矛盾。我们利用所有发现制作了 ProLong，一个新的最先进的长上下文语言模型。我们公开发布了所有代码、数据和模型，并希望我们的发现能推动长上下文语言模型的研究和应用。

Limitations

Although we try to ablate the major components of our training recipe, due to resource limitations, we cannot exhaust all aspects, such as the optimization hyperparameters and additional data mixtures. We also limit ourselves to the 10B-scale regime and the Llama-3 models, which may limit the generalizability of our findings and recipe. Another concern is that we are overfitting to the tasks chosen for model development-however, we do not directly train on those datasets and guiding model development with benchmark tasks has become a common practice in pre-trained LM development. We also show that our final recipe and model perform well on an additional evaluation dataset, NoCha.

尽管我们尝试消除训练方案的主要组成部分，但由于资源限制，我们无法涵盖所有方面，例如优化超参数和额外的数据混合。我们还将自己限制在 10B 规模的范围和 Llama-3 模型，这可能会限制我们发现和方案的普遍适用性。另一个问题是我们可能过拟合于模型开发所选择的任务——然而，我们并没有直接在这些数据集上进行训练，并且通过基准任务指导模型开发已成为预训练语言模型开发中的一种常见做法。我们还展示了我们的最终方案和模型在额外评估数据集 NoCha 上表现良好。

Appendix Experiment Details

A.1 Evaluation

Table 13: The details for our long-context evaluation following HELMET (Yen et al., 2024b).
我们根据 HELMET (Yen et al., 2024b) 进行的长上下文评估的详细信息。

Category	Metrics	Tasks and Datasets
Recall	SubEM	Given a randomly-generated long JSON file and a key, retrieve the corresponding value (Liu et al., 2024b).
RAG	SubEM	Given a question and many retrieved Wikipedia documents (shuf- fled), answer the question (Liu et al., 2024b). Datasets: NaturalQues- tion (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), and PopQA (Mallen et al., 2023).
Re-rank	nDCG@10	Given a query and many retrieved documents (shuffled), re-rank the top-10 documents. Datasets: MSMARCO (Bajaj et al., 2016).
ICL	Accuracy	Datasets selected from Bertsch et al. (2024): TREC coarse, TREC fine (Hovy et al., 2001), NLU (Liu et al., 2021), Bank- ing77 (Casanueva et al., 2020), and Clinc-150 (Larson et al., 2019).
QA	GPT-4o score	Given a book, answer the question. Datasets (#tokens): Narra- tiveQA (medium: 73K; max: 518K; Kočiský et al., 2018).
Summ.	GPT-4o score	Summarize a given legal document. Datasets (#tokens): Multi- LexSum (medium: 90K; max: 5M; Shen et al., 2022)

Table 13 shows all the datasets we used for the long-context evaluation from HELMET (Yen et al., 2024b). Note that we did not use all the datasets from HELMET for efficiency reasons and we also do not want to overfit to HELMET. We highlight some of the evaluation protocol improvements that HELMET implemented compared to previous benchmarks here:
表 13 显示了我们用于 HELMET (Yen et al., 2024b) 的长上下文评估的所有数据集。请注意，由于效率原因，我们并未使用 HELMET 的所有数据集，并且我们也不希望对 HELMET 进行过拟合。我们在此强调 HELMET 相较于之前基准的一些评估协议改进:

- Sufficient context lengths and fine-grained control. HELMET can evaluate models at a context length of 128 K tokens and beyond. The evaluation protocol also allows for reporting results at different lengths, giving developers fine-trained controls for different needs of long contexts.

  - 足够的上下文长度和细粒度控制。HELMET 可以在 128 K 个标记及以上的上下文长度下评估模型。评估协议还允许在不同长度下报告结果，为开发者提供了针对长上下文不同需求的细致控制。

- Better synthetic recall tasks. As shown in HELMET, needle-in-a-haystack (Kamradt, 2024) is mostly saturated because of its simplicity—the model only needs to find a needle in some irrelevant context. We instead use the more challenging JSON KV task, first proposed in Liu et al. (2024b) and included in HELMET, where the model is required to find the corresponding value to a given key among a large JSON file.

  - 更好的合成召回任务。如 HELMET 所示，针在干草堆中 (Kamradt, 2024) 由于其简单性而大多饱和——模型只需在一些无关的上下文中找到一根针。我们则使用更具挑战性的 JSON KV 任务，该任务首次在 Liu et al. (2024b) 中提出并包含在 HELMET 中，模型需要在一个大型 JSON 文件中找到与给定键对应的值。

- Using class-balanced demonstrations and abstract labels for ICL. To disentangle models' ability of learning from demonstrations from their pre-training bias of the task or the dataset label distribution (Pan et al., 2023), HELMET samples the same number of demonstrations for each class and uses number labels $(1, 2, \ldots)$ instead of natural-language labels (e.g., location, description, ...).

  - 使用类别平衡的示例和抽象标签进行 ICL。为了将模型从示例中学习的能力与其在任务或数据集标签分布上的预训练偏差 (Pan et al., 2023) 解耦，HELMET 为每个类别抽样相同数量的示例，并使用数字标签 $(1, 2, \ldots)$ 而不是自然语言标签 (例如，位置、描述等)。

- Model-based evaluation for long-context QA and summarization. Instead of using traditional metrics like ROUGE (which has shown to be poorly indicative of the real model performance: Deutsch & Roth, 2021; Deutsch et al., 2022; Goyal et al., 2023; Chang et al., 2024), HELMET uses model-based evaluations to compare the reference answer and the model output. For QA, HELMET uses GPT-4o to score the model output given the question and the reference answer at a 0-3 scale. For summarization, HELMET takes a similar approach as Zhang & Bansal (2021); Gao et al. (2023): it first uses GPT-4o to decompose the reference summary into atomic claims; then it uses GPT-4o to check whether each reference atomic claim is covered by the model output (recall) and whether each sentence in the model output is covered by the reference summary (precision). Yen et al. (2024b) show that the model-based evaluation correlates with human perceptions significantly better than traditional metrics.

  - 基于模型的长上下文 QA 和摘要评估。HELMET 不使用传统指标如 ROUGE(已显示对真实模型性能的指示性较差:Deutsch & Roth, 2021; Deutsch et al., 2022; Goyal et al., 2023; Chang et al., 2024)，而是使用基于模型的评估来比较参考答案和模型输出。对于 QA，HELMET 使用 GPT-4o 在 0-3 的尺度上对给定问题和参考答案的模型输出进行评分。对于摘要，HELMET 采用与 Zhang & Bansal (2021); Gao et al. (2023) 类似的方法: 首先使用 GPT-4o 将参考摘要分解为原子声明；然后使用 GPT-4o 检查模型输出是否覆盖每个参考原子声明 (召回) 以及模型输出中的每个句子是否被参考摘要覆盖 (精确度)。Yen et al. (2024b) 显示，基于模型的评估与人类感知的相关性显著优于传统指标。

A.2 Data processing

Data sources. We list all the data sources we have explored in our ablations and main experiments here: the Stack (Kocetkov et al., 2023), SlimPajama (Together, 2023; Soboleva et al., 2023), FineWeb (we use the 2023-50 snapshot), FineWeb-Edu (we use a random sample) (Penedo et al., 2024), Tulu-v2 (Ivison et al., 2023), OpenWebMath (Paster et al., 2024), textbooks (Chevalier et al., 2024), and Dolma (Soldaini et al., 2024). The Books, StackExchange, and ArXiv data are from SlimPajama. The Wikipedia data are from Dolma.

数据来源。我们在此列出所有在消融实验和主要实验中探索的数据来源:Stack (Kocetkov et al., 2023)、SlimPajama (Together, 2023; Soboleva et al., 2023)、FineWeb (我们使用 2023-50 快照)、FineWeb-Edu (我们使用随机样本) (Penedo et al., 2024)、Tulu-v2 (Ivison et al., 2023)、OpenWebMath (Paster et al., 2024)、教科书 (Chevalier et al., 2024) 和 Dolma (Soldaini et al., 2024)。Books、StackExchange 和 ArXiv 数据来自 SlimPajama。Wikipedia 数据来自 Dolma。

Data filtering and packing. For the short training data and the SFT data, we randomly sample and concatenate the documents or conversations into 64 K chunks. The last document for each chunk is truncated. The truncated part is used as the beginning for the next chunk for the short training data but is discarded for the SFT data. For the long-context training data, we filter out the documents that are shorter than 64 K ; we do the same for the 512 K setting, while making sure that the 64 K documents packed to 512 K length are distinct from the 512 K documents.

数据过滤和打包。对于短训练数据和 SFT 数据，我们随机抽样并将文档或对话连接成 64 K 块。每个块的最后一个文档被截断。截断部分用于短训练数据的下一个块的开头，但对于 SFT 数据则被丢弃。对于长上下文训练数据，我们过滤掉短于 64 K 的文档；对于 512 K 设置，我们也这样做，同时确保打包到 512 K 长度的 64 K 文档与 512 K 文档是不同的。

Final data mixture. For 512K length, we use a mix of 64K and 512K long data. For the ratio of 64K/512K data, we choose 50%/50% for code and 83%/17%, which are roughly chosen according to the natural availability of very long data, i.e., there are relatively fewer books of length 512K than code repositories. One benefit of retaining 64K-long documents is that we can process these without sequence parallelism and the associated communication overhead. We use a slightly different long data mixture in our ablations (Table 5) and our main ProLong experiment (Table 9). For the final model, we mix 3% textbooks into the long-context training data. The textbooks are open-source resources from libretexts.org, collected and made available by Chevalier et al. (2024). We pre-process the data by concatenating chapters from the same text books, as well as books from the same subject areas. This results in extremely long sequences which we pack into contexts of either 64K or 512K tokens. Though we do not have an ablation for adding this data due to limited resources, we believe that it should have a slight positive effect to the final model performance as textbooks are highly educational long-context data.

最终数据混合。对于 512K 长度，我们使用 64K 和 512K 长数据的混合。对于 64K/512K 数据的比例，我们选择代码的 50%/50% 和 83%/17%，这些比例大致是根据非常长数据的自然可用性选择的，即，长度为 512K 的书籍相较于代码库相对较少。保留 64K 长文档的一个好处是我们可以在没有序列并行性和相关通信开销的情况下处理这些文档。我们在消融实验 (表 5) 和主要 ProLong 实验 (表 9) 中使用了略有不同的长数据混合。对于最终模型，我们将 3% 的教科书混入长上下文训练数据中。这些教科书是来自 libretexts.org 的开源资源，由 Chevalier et al. (2024) 收集并提供。我们通过连接来自同一教科书的章节以及来自同一学科领域的书籍来预处理数据。这导致了极长的序列，我们将其打包成 64K 或 512K token 的上下文。尽管由于资源有限，我们没有对添加这些数据进行消融实验，但我们相信这应该对最终模型性能有轻微的积极影响，因为教科书是高度教育性的长上下文数据。

Table 14: % Proportion of long documents for the short data components used in Table 6. These statistics are computed after packing and truncation and therefore correspond to the document lengths as seen by the model. We highlight that the proportion of documents beyond 32K is below 1% for ShortMix.
表 6 中使用的短数据组件的长文档比例。这些统计数据是在打包和截断后计算的，因此对应于模型所见的文档长度。我们强调，ShortMix 中超过 32K 的文档比例低于 1%。

	>4K	>8K	>16K	>32K
FineWeb	1.4	0.3	0.1	0.0
FineWeb-Edu	2.8	0.8	0.2	0.0
Wikipedia	1.6	0.4	0.0	0.0
Tulu-v2	0.0	0.0	0.0	0.0
StackExchange	0.6	0.1	0.0	0.0
ArXiv	85.7	64.0	30.3	7.6
OpenWebMath	11.1	4.3	1.2	0.3
ShortMix	10.9	7.2	3.2	0.8
SlimPajama	11.3	7.4	4.9	3.2
FineWeb-Edu	2.8	0.8	0.2	0.0
DCLM-Baseline	4.9	1.7	0.4	0.1

A.3 Implementation details **实施细节**

Technical stack. We use various open-source packages and tools for the ProLong training and evaluation. We use PyTorch (Paszke et al., 2019) and Hugging Face transformers (Wolf et al., 2020) for the model training. We use mosaic-streaming (Mosaic ML, 2022) for loading and mixing the data and FlashAttention 2 (Dao, 2024) for efficient attention implementation. We implement sequence parallelism based on DeepSpeed-Ulysses (Jacobs et al., 2023) across groups of 8 GPUs on the same node. We only perform distributed attention if it is necessary, i.e., only on sequences of 512K length. For long-context evaluation, we use HELMET (Yen et al., 2024b) and for short-context evaluation, we use lm-eval-harness (Gao et al., 2021).

技术栈。我们使用各种开源软件包和工具进行 ProLong 的训练和评估。我们使用 PyTorch (Paszke et al., 2019) 和 Hugging Face transformers (Wolf et al., 2020) 进行模型训练。我们使用 mosaic-streaming (Mosaic ML, 2022) 来加载和混合数据，并使用 FlashAttention 2 (Dao, 2024) 进行高效的注意力实现。我们基于 DeepSpeed-Ulysses (Jacobs et al., 2023) 在同一节点的 8 个 GPU 组上实现序列并行。只有在必要时，我们才执行分布式注意力，即仅在长度为 512K 的序列上进行。对于长上下文评估，我们使用 HELMET (Yen et al., 2024b)，而对于短上下文评估，我们使用 lm-eval-harness (Gao et al., 2021)。

Attention and batching. Since we do document masking in attention (§6), we use the variable-length attention implementation from FlashAttention 2 (Dao, 2024) to speed up long-context training: for sequences that are concatenations of multiple short documents, instead of computing the full attention with masking, we instead compute the attention for each individual document. Since the complexity of attention is quadratic to the sequence length, this improves the training speed. However, the improvement is negligible in a distributed training setting with FSDP, since GPUs processing short sequence batches have to wait on other GPUs processing long sequences. We therefore implement a smart batching algorithm: In our setting, a gradient step usually consists of multiple gradient accumulation steps, where each device processes a smaller minibatch. We sort all the minibatches per training step by the sum of the squared lengths of documents in the sequence. This leads to more balanced sequence lengths across the GPUs and effective speedups, as can be seen in Table 15, without affecting the gradient updates or loss during training. However, the efficiency gains are diminished when training with more GPUs, as this reduces the number of gradient accumulation steps.

注意力和批处理。由于我们在注意力中进行文档掩蔽 (§6)，我们使用 FlashAttention 2 (Dao, 2024) 的可变长度注意力实现来加速长上下文训练: 对于多个短文档的连接序列，我们不计算带掩蔽的完整注意力，而是计算每个单独文档的注意力。由于注意力的复杂度与序列长度的平方成正比，这提高了训练速度。然而，在使用 FSDP 的分布式训练环境中，改进是微不足道的，因为处理短序列批次的 GPU 必须等待处理长序列的其他 GPU。因此，我们实现了一种智能批处理算法: 在我们的设置中，一个梯度步骤通常由多个梯度累积步骤组成，每个设备处理一个较小的迷你批次。我们根据序列中文档长度的平方和对每个训练步骤的所有迷你批次进行排序。这导致 GPU 之间的序列长度更加平衡，并有效加速，如表 15 所示，而不会影响训练过程中的梯度更新或损失。然而，当使用更多 GPU 进行训练时，效率提升会减小，因为这减少了梯度累积步骤的数量。

Table 15: Throughput per device of our ablation runs from Table 20, when training with 8 Nvidia H100 GPUs with FSDP. Our strategy of reordering minibatches is important for realizing the speed

benefits from variable-length attention.
表 15: 在使用 8 个 Nvidia H100 GPU 和 FSDP 进行训练时，我们的消融实验运行的每个设备的吞吐量，来自表 20。我们重新排序迷你批次的策略对于实现可变长度注意力的速度优势至关重要。

	Throughput (tokens/s/GPU)
64K full attention	2770
Variable-length attention	$2780_{(+0.4\%)}$
+ Minibatch reordering	3095(+11.7%)

Token-averaged loss. We found that in the SFT stage, the distribution of the training tokens (in SFT, the tokens from the instructions are masked out and the models are only trained on the responses) on each GPU device can be extremely imbalanced, especially when there is synthetic data (most tokens in a synthetic data instance are from the instruction). Conventional all-reduce loss in distributed training averages over the sequences instead of valid tokens, which skews the optimization and also our control over the domain proportions. Instead, we change the all-reduce loss to be the average over all valid training tokens. Bai et al. (2024a) implements their SFT loss in a similar way.

基于 token 的平均损失。我们发现，在 SFT 阶段，训练 token 的分布 (在 SFT 中，指令中的 token 被屏蔽，模型仅在响应上进行训练) 在每个 GPU 设备上可能极为不平衡，特别是在存在合成数据时 (合成数据实例中的大多数 token 来自指令)。传统的分布式训练中的全归约损失是对序列进行平均，而不是有效 token，这扭曲了优化过程，也影响了我们对领域比例的控制。相反，我们将全归约损失更改为对所有有效训练 token 的平均值。Bai 等人 (2024a) 以类似的方式实现了他们的 SFT 损失。

A.4 The ablation setting **消融设置**

For all our ablations, unless specified, we train the base model of Llama-3-8B (instead of Instruct) on a 64 K sequence length for 5 B tokens, with the same hyperparameters as specified in Table 9. We choose this context length, as it is the highest power of 2 value for which we can train without sequence parallelism. By default, we use the same training data as the 64K ProLong setting, except that we remove the textbooks and use the ShortMix proportions in Table 5. For SFT, we use the same settings as specified in Table 9.

对于我们所有的消融实验，除非另有说明，我们在 64 K 序列长度上以与表 9 中指定的相同超参数训练 Llama-3-8B 的基础模型 (而不是 Instruct)，用于 5 B token。我们选择这个上下文长度，因为这是我们可以在没有序列并行性的情况下进行训练的最大 2 的幂值。默认情况下，我们使用与 64K ProLong 设置相同的训练数据，除了我们去除了教科书，并使用表 5 中的 ShortMix 比例。对于 SFT，我们使用与表 9 中指定的相同设置。

A.5 Generating synthetic SFT data A.5 **生成合成** SFT **数据**

We prompt Llama-3-8B-Instruct to generate the synthetic data and Table 16 shows the prompt we used for generating the synthetic QA data for books. We also write predefined templates and randomly sample one for each synthetic instance to increase the diversity, and Table 17 provides some examples.

我们提示 Llama-3-8B-Instruct 生成合成数据，表 16 显示了我们用于生成书籍合成 QA 数据的提示。我们还编写了预定义模板，并为每个合成实例随机抽取一个以增加多样性，表 17 提供了一些示例。

Table 16: Prompts for generating synthetic QA data.

Given the following snippet of a book, ask a relevant question and provide the answer. The question and the answer should follow the following rules:
(1) The question should be specific enough that it can only be answered with the snippet. The question should also be interesting and intellectual enough that a curious reader of the book would ask about it.
(2) The question and the answer should be comprehensible given just the whole book

without highlighting the snippet. With that being said, the question should NOT refer to the snippet directly (e.g., do NOT say things like "Question: given the conversation in the snippet, what ..."). The answer also should not mention "the snippet ..." explicitly (assuming that the snippet is never provided), but it can copy the snippet content as a reference when answering the question.
(3) The answer should be concise but also should provide references to the book when needed. For example, Wellington Yueh betrayed the Atreides, as the book mentioned, , ... ".
*** Start of the snippet ***
{sampled snippet}
*** End of the snippet ***
Before generating the question and the answer, first reason about what this snippet is about. In your generation, stick to the following format:
Reasoning: this snippet is about ...
Question: ...
Answer: ...

Table 17: Examples for question prompts and templates used for generating diverse synthetic QA data. We sample one question prompt and one template each time and combine them with the documents and the generated QA pairs to form a synthetic training example.

表 17: 用于生成多样化合成 QA 数据的问题提示和模板示例。我们每次抽取一个问题提示和一个模板，并将它们与文档和生成的 QA 对结合，形成一个合成训练示例。

Example question prompts for synthetic QA data

合成 QA 数据的示例问题提示

Given the document, please answer the question.

根据文档，请回答问题。

Here is a piece of text; answer the following question based on it.

这里有一段文本；请根据它回答以下问题。

Please answer the question using the provided content.

请使用提供的内容回答问题。

Based on the given passage, respond to the question.

根据给定的段落，回应问题。

Read the snippet and answer the question that follows.

阅读摘录并回答随后的问题。

Using the provided text, answer the following question.

使用提供的文本，回答以下问题。

Example templates for combining questions, answers, and contexts for synthetic QA data

合成 QA 数据的问题、答案和上下文的示例模板

```
        {prompt} \n
n{documents} \n
nQuestion: {question}
    {prompt} \n
n==== document starts ==== \n{documents} \n==== document ends
       === \n
nQuestion: {question}
       {prompt} \n
n{documents} \n
n{question}
         {prompt} Question: {question} \n
n{documents}
           {prompt} {question} \n
n{documents}
           {prompt} \n
n{question} \n
n{documents}
```

B More Ablations **更多消融实验**

B.1 Position extrapolation **位置外推**

Xiong et al. (2023); emozilla (2023) show that changing the RoPE frequency base to a larger value in continual long-context pre-training or in inference time can improve the long-context performance. emozilla (2023) suggests that one should scale the frequency base by a factor of $t^{\frac{d}{d-2}}$, where t is the ratio between the target sequence length and the original LM length, and d is the attention head dimension.

Xiong 等人 (2023); emozilla (2023) 表明，在持续的长上下文预训练或推理时，将 RoPE 频率基数更改为更大的值可以改善长上下文性能。emozilla (2023) 建议应将频率基数按 $t^{\frac{d}{d-2}}$ 的比例缩放，其中 t 是目标序列长度与原始语言模型长度之间的比率，d 是注意力头维度。

We conduct ablation studies, at both 64 K (same as our standard ablation setting as specified in §A.4) and 512K (starting from ProLong-64K and training with the 512K data mixture for 5B tokens) sequence lengths, on what frequency bases we should use. Table 18 and Table 19 show the results. We first see that using the original 500,000 frequency base from Llama-3 leads to significant performance degradation. While dynamic NTK suggests 4×10^6 , we find that further scaling it to 8×10^6 leads to better performance. Similar, we see that when scaling the 64 K model to 512 K , while dynamic NTK suggests a 64×10^6 frequency base, much larger frequency bases $\left(128 \times 10^6 \text{ and } 256 \times 10^6\right)$ lead to better performance. We use 8×10^6 for 64 K and 128×10^6 for 512 K for our final ProLong models.

我们在 64 K (与 §A.4 中指定的标准消融设置相同) 和 512K(从 ProLong-64K 开始，并使用 512K 数据混合训练 5B token) 序列长度上进行消融研究，以确定我们应使用的频率基数。表 18 和表 19 显示了结果。我们首先看到，使用 Llama-3 的原始 500,000 频率基数会导致显著的性能下降。虽然动态 NTK 建议 4×10^6 ，但我们发现进一步将其缩放到 8×10^6 会带来更好的性能。同样，我们看到当将 64 K 模型缩放到 512 K 时，虽然动态 NTK 建议使用 64×10^6 频率基数，但更大的频率基数 $\left(128 \times 10^6 \text{ 和 } 256 \times 10^6\right)$ 会带来更好的性能。我们在最终的 ProLong 模型中使用 8×10^6 作为 64 K 和 128×10^6 作为 512 K 。

Table 18: Ablation study on RoPE frequency base at a maximum training length of 64K. Dynamic NTK (emozilla,2023) roughly suggests to use 4m as the frequency base.

表 18: 在最大训练长度为 64K 的情况下，RoPE 频率基的消融研究。动态 NTK (emozilla,2023) 大致建议使用 4m 作为频率基。

RoPE Base ($\times 10^6$)	Long-Context							Short-Context
	Recall	RAG	Re-rank	ICL	QA	Summ.	Avg.	Avg.
0.5	25.8	37.0	4.4	73.8	17.5	16.3	29.1	65.0
4.0	81.3	47.8	18.2	76.5	31.8	36.3	48.7	65.3
8.0	96.0	54.9	29.4	73.9	35.7	37.9	54.6	65.5

RoPE 基础 ($\times 10^6$)	长上下文							短上下文
	召回	检索增强生成	重新排序	ICL	问答	总结	平均	平均
0.5	25.8	37.0	4.4	73.8	17.5	16.3	29.1	65.0
4.0	81.3	47.8	18.2	76.5	31.8	36.3	48.7	65.3
8.0	96.0	54.9	29.4	73.9	35.7	37.9	54.6	65.5

Table 19: Ablation study on RoPE frequency base at a maximum training length of 512K. Dynamic NTK (emozilla,2023) roughly suggests to use 64×10^6 as the frequency base.

表 19: 在最大训练长度为 512K 的情况下，RoPE 频率基的消融研究。动态 NTK (emozilla,2023) 大致建议使用 64×10^6 作为频率基。

RoPE Base ($\times 10^6$)	Long-Context							Short-Context
	Recall	RAG	Re-rank	ICL	QA	Summ.	Avg.	Avg.
64	98.8	57.8	30.4	82.2	38.2	38.3	57.6	68.3
128	98.8	57.4	30.7	80.0	40.4	38.8	57.7	68.6
256	98.8	56.8	33.8	79.8	37.9	39.7	57.8	68.4

RoPE 基础 ($\times 10^6$)	长上下文							短上下文
	召回	检索增强生成	重新排序	ICL	问答	总结	平均	平均
64	98.8	57.8	30.4	82.2	38.2	38.3	57.6	68.3
128	98.8	57.4	30.7	80.0	40.4	38.8	57.7	68.6
256	98.8	56.8	33.8	79.8	37.9	39.7	57.8	68.4

B.2 Document masks **文档掩码**

We experiment whether to use document masks in attention in Table 20. Standard training concatenates multiple short documents into a single sequence (in our case, a 64K sequence), uses a special token to separate documents, and performs full attention over the whole sequence. When the document masks are used, we do not allow the attention to cross the document boundaries. We find that using document masks in continual long-context training leads to both better long-context results and short-context performance. For all our other ablations and the main experiment, we use document masks.

我们实验是否在表 20 中使用文档掩码进行注意力机制。标准训练将多个短文档连接成一个单一序列 (在我们的案例中是一个 64K 序列)，使用特殊标记分隔文档，并对整个序列执行完全注意力。当使用文档掩码时，我们不允许注意力跨越文档边界。我们发现，在持续的长上下文训练中使用文档掩码可以带来更好的长上下文结果和短上下文性能。对于我们所有其他的消融实验和主要实验，我们都使用文档掩码。

Table 20: Impact of using document masks in attention.

表 20: 使用文档掩码对注意力的影响。

Attention	Long-Context							Short-Context
	Recall	RAG	Re-rank	ICL	QA	Summ.	Avg.	Avg.
No doc masks	97.4	53.6	20.4	76.6	37.2	36.3	53.6	64.9
Document masks	96.0	54.9	29.4	73.9	35.7	37.9	54.6	65.5

注意力	长上下文							短上下文
	召回	检索增强生成	重新排序	ICL	问答	总结	平均	平均
无文档掩码	97.4	53.6	20.4	76.6	37.2	36.3	53.6	64.9
文档掩码	96.0	54.9	29.4	73.9	35.7	37.9	54.6	65.5

B.3 Initialization **初始化**

We use the base model for Llama-3-8B as the initialization for all our ablations to make sure the findings are generalizable and are not confounded by the Llama instruction tuning. However, for our final ProLong model, we use Llama-3-8B-Instruct as the initialization to achieve the best performance. We see in Table 21 (using the ablation setting from §A.4) that using Llama-3-8B-Instruct as the initialization achieves slightly better long-context performance and much stronger short-context performance.

我们使用 Llama-3-8B 的基础模型作为所有消融实验的初始化，以确保研究结果具有普遍性，并且不受 Llama 指令调优的干扰。然而，对于我们的最终 ProLong 模型，我们使用 Llama-3-8B-Instruct 作为初始化，以实现最佳性能。我们在表 21 中看到 (使用 §A.4 中的消融设置)，使用 Llama-3-8B-Instruct 作为初始化可以实现稍微更好的长上下文性能和更强的短上下文性能。

Table 21: Differences of using the base Llama-3-8B model vs. Llama-3-8B-Instruct.

表 21: 使用基础 Llama-3-8B 模型与 Llama-3-8B-Instruct 的差异。

Base Model	Long-Context	Short-Context					
	Avg.	HellaS.	MMLU	ARC-c	WG	GSM8K	Avg.
Llama-3-8B-Base	54.6	81.6	65.3	58.0	76.2	46.6	65.5
Llama-3-8B-Instruct	55.0	80.8	66.1	58.5	75.6	57.7	67.7

基础模型	长上下文	短上下文					
	平均	HellaS.	MMLU	ARC-c	WG	GSM8K	平均
Llama-3-8B-基础	54.6	81.6	65.3	58.0	76.2	46.6	65.5
Llama-3-8B-指令	55.0	80.8	66.1	58.5	75.6	57.7	67.7

B.4 Instruction-tuning datasets **指令调优数据集**

Initialized from the ProLong base model, we experiment with different public, short-context SFT datasets. All runs use the same SFT hyperparameters as specified in Table 9. Table 22 shows that using UltraChat leads to the best overall results. Note that this does not necessarily mean that UltraChat is the best SFT dataset for all base models or applications.

我们从 ProLong 基础模型初始化，实验不同的公共短上下文 SFT 数据集。所有实验使用与表 9 中指定的相同 SFT 超参数。表 22 显示使用 UltraChat 得到最佳整体结果。请注意，这并不一定意味着 UltraChat 是所有基础模型或应用的最佳 SFT 数据集。

Table 22: Ablations on using different short-context SFT datasets. We report the 32K-and- 64K-averaged performance except tasks marked with † , which are evaluated at 512K for stress testing.

表 22: 使用不同短上下文 SFT 数据集的消融实验。我们报告 32K 和 64K 平均性能，除了标记为 † 的任务，这些任务在 512K 下进行压力测试评估。

SFT Data	Long-Context						
	Recall †	RAG	Re-rank	ICL	QA*	Summ. †	Avg.
UltraChat	65.7	58.1	38.5	80.3	49.7	42.1	55.7
Tulu v2	61.5	45.4	25.1	81.8	40.4	40.3	49.1
ShareGPT	40.5	47.5	26.7	79.6	42.7	34.4	45.2

SFT 数据	长上下文						
	回忆 †	RAG	重新排序	ICL	问答 *	总结 †	平均值
超聊天	65.7	58.1	38.5	80.3	49.7	42.1	55.7
Tulu v2	61.5	45.4	25.1	81.8	40.4	40.3	49.1
分享 GPT	40.5	47.5	26.7	79.6	42.7	34.4	45.2

C Full Evaluation Results 完整评估结果

We provide all the evaluation results, including specific numbers for each dataset and evaluation length, at https://shorturl.at/pULbx.

我们提供所有评估结果，包括每个数据集和评估长度的具体数字，网址为 https://shorturl.at/pULbx。

D Examples

Table 23 shows some long-document QA examples with ProLong, Llama-3.1, and GPT-4o.

表 23 显示了一些使用 ProLong、Llama-3.1 和 GPT-4o 的长文档问答示例。

Table 23: Long-document QA examples with ProLong (512K), Llama-3.1-8B-Instruct (128K), and GPT-4o (via the ChatGPT interface). Note that it is very likely that the ChatGPT-version GPT-4o uses a retrieval component for long documents, hence it can process contexts beyond its 128K context window. We see that ProLong can answer questions effectively with a 300K document while both Llama-3.1 and GPT-4o fail. On a shorter document (Llama-3.1 paper), ProLong can understand very technical questions and answer them accurately.

表 23: 使用 ProLong (512K)、Llama-3.1-8B-Instruct (128K) 和 GPT-4o (通过 ChatGPT 接口) 的长文档问答示例。请注意，ChatGPT 版本的 GPT-4o 很可能使用了检索组件来处理长文档，因此它可以处理超出其 128K 上下文窗口的上下文。我们看到 ProLong 能够有效回答 300K 文档中的问题，而 Llama-3.1 和 GPT-4o 则失败。在较短的文档 (Llama-3.1 论文) 中，ProLong 能够理解非常专业的问题并准确回答。