

LONGLORA：长上下文大型语言模型的高效微调

Yukang Chen¹ **Shengju Qian**¹ **Haotian Tang**² **Xin Lai**¹

Zhijian Liu² **Song Han**^{2,3} **Jiaya Jia**¹

¹CUHK

²MIT

³NVIDIA

ABSTRACT

我们提出了 LongLoRA，这是一种有效的微调方法，可以扩展预训练大型语言模型 (LLM) 的上下文大小，且计算成本有限。通常，训练具有长上下文大小的法语硕士的计算成本很高，需要大量的训练时间和 GPU 资源。例如，在 8192 的上下文长度上进行训练需要比 2048 少 16 倍的自注意力层计算成本。在本文中，我们从两个方面加速了 LLM 的上下文扩展。一方面，虽然推理过程中需要密集的全局注意力，但通过稀疏的局部注意力可以有效且高效地对模型进行微调。所提出的转移稀疏注意力 (S₂-Attn) 有效地实现了上下文扩展，从而节省了大量的计算量，并且具有与普通注意力微调相似的性能。特别是，它在训练中只需两行代码即可实现，而在推理中是可选的。另一方面，我们重新审视上下文扩展的参数有效微调机制。值得注意的是，我们发现用于上下文扩展的 LoRA 在可训练嵌入和标准化的前提下效果很好。LongLoRA 将这种改进的 LoRA 与 S₂-Attn 结合起来。LongLoRA 在 Llama2 模型上从 7B/13B 到 70B 的各种任务上展示了强有力的实证结果。LongLoRA 将 Llama2 7B 从 4k 上下文扩展到 100k，或在单个 8 × A100 机器上将 Llama2 70B 扩展到 32k。LongLoRA 扩展了模型的上下文，同时保留了其原始架构，并且与大多数现有技术兼容，例如 Flash-Attention2。此外，我们进一步使用 LongLoRA 和我们的长指令跟踪 LongAlpaca 数据集进行监督微调。我们所有的代码、模型、数据集和演示都可以在 github.com/dvlab-research/LongLoRA 上找到。

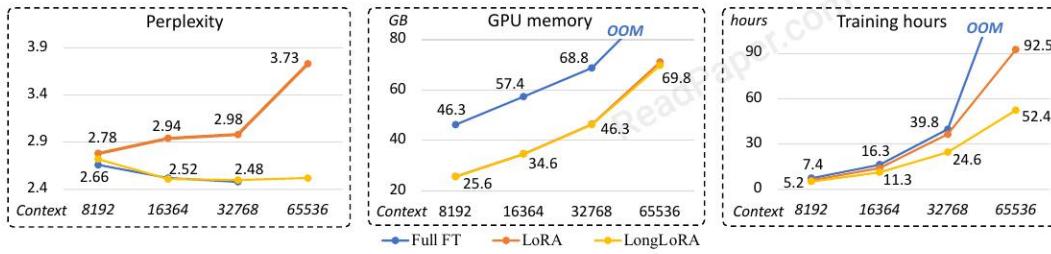


图 1：LongLoRA 缩小了传统 LoRA 与完全微调之间的精度差距，同时仍保持比完全微调低 1.8 倍的内存成本。此外，LongLoRA 通过 S₂-Attn 将 LoRA 的训练速度提高了 1.8 倍。Llama2-7B 通过 Flash-Attention2 (Dao, 2023) 和 DeepSpeed (Rasley 等人, 2020) 第 2 阶段微调到各种上下文长度，并在证明堆 (Azerbayev 等人, 2022) 测试集上进行评估困惑。

1 简介

大型语言模型 (LLM) 通常使用预定义的上下文大小进行训练，例如 LLaMA 的 2048 个标记 (Touvron 等人, 2023a) 和 Llama2 的 4096 个标记 (Touvron 等人, 2023b)。

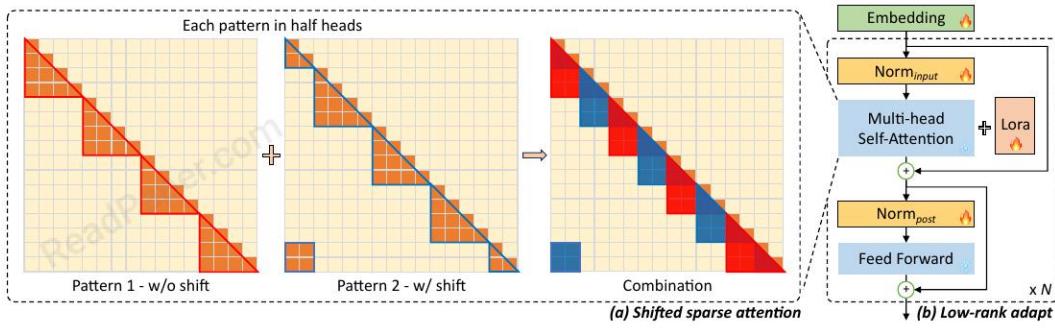


图 2: LongLoRA 概述。我们在微调期间引入了转移稀疏注意力 (S 2 -Attn)。训练后的模型在推理时保留了原始标准的自注意力。除了在线性层中训练 LoRA 权重之外, LongLoRA 还进一步使嵌入层和归一化层变得可训练。此扩展对于上下文扩展至关重要, 并且仅引入了最少量的附加可训练参数。

! 可训练 — 冻结

然而, 预定义的大小限制了法学硕士在许多应用中的应用, 例如总结长文档或回答长问题。为了解决这一限制, 最近的一些工作 (Chen et al., 2023; Tworkowski et al., 2023; Mohtashami & Jaggi, 2023) 对法学硕士进行了训练或微调以适应更长的背景。然而, 使用长序列从头开始训练 LLM 会带来计算挑战, 并且对现有预训练的 LLM 进行微调也相当昂贵。例如, Position Interpolation (Chen 等人, 2023) 使用 32 个 A100 GPU 将 LLaMA 模型从 2k 上下文扩展到 8k, 并使用 128 个 A100 GPU 进行更长的上下文微调。FOT (Tworkowski 等人, 2023) 使用 32 个 TPU 进行标准 Transformer 训练, 使用 128 个 TPU 进行 LongLLaMA。这些计算资源对于普通研究人员来说通常是难以承受的, 这自然会引发我们的疑问: 我们能否有效地扩展法学硕士的上下文窗口?

一种直接的方法是通过低秩适应 (LoRA) 微调预训练的 LLM (Hu 等人, 2022)。LoRA 通过利用低秩矩阵修改自注意力块中的线性投影层, 这通常很有效并且减少了可训练参数的数量。然而, 我们的实证研究结果表明, 以这种方式训练长上下文模型既不够有效也不高效。就有效性而言, 简单的低秩自适应会导致长上下文扩展中的高度困惑, 如表 2 所示。将秩增加到更高的值 (例如, 秩 = 256) 并不能缓解此问题。在效率方面, 无论是否采用 LoRA, 随着上下文大小的扩展, 计算成本都会急剧增加, 这主要是由于标准的自注意力机制 (Vaswani et al., 2017)。如图 1 所示, 即使使用 LoRA, 当上下文窗口扩大时, 标准 Llama2 模型的训练时间也会大幅增加。

在这项工作中, 我们引入了 LongLoRA, 这是一种有效的微调方法, 可以扩展预训练的 LLM 的上下文窗口, 例如 Llama2 (Touvron 等人, 2023b)。LoRA (Hu et al., 2022) 使用低秩权重更新来近似完全微调。同样, 我们发现短注意力也能够在训练过程中近似长上下文。我们提出转移稀疏注意力 (S 2 -Attn) 作为标准自注意力的有效替代品。如图 2 所示, 我们将上下文长度分为几组, 并在每个组中分别进行注意。在半注意力头中, 我们将标记移动一半的组大小, 这确保了相邻组之间的信息流。例如, 我们使用组大小为 2048 的 S 2 -Attn 来近似训练总共 8192 个上下文长度。这与 Swin Transformer 具有较高的精神层面 (Liu et al., 2021)。

通过 S 2 -Attn 微调的模型在推理过程中保留了原始的注意力架构。这有利于大多数现有的优化和基础设施。常见法学硕士的技术也可以应用于我们的。例

如, Flash-Attention2 (Dao et al., 2022; Dao, 2023) 在训练和推理时间上都与我们的方法兼容。其背后的原因是短注意力类似于法学硕士预训练阶段的注意力方案。其他有效的注意力, 例如扩张或稀疏注意力, 与标准风格有很大差距, 并且不像我们的那样工作得很好, 如表 6 所示。

我们凭经验表明, 可学习的嵌入和归一化层是解锁长上下文 LoRA 微调的关键, 如表 2 所示。嵌入和归一化层占用的空间很小。

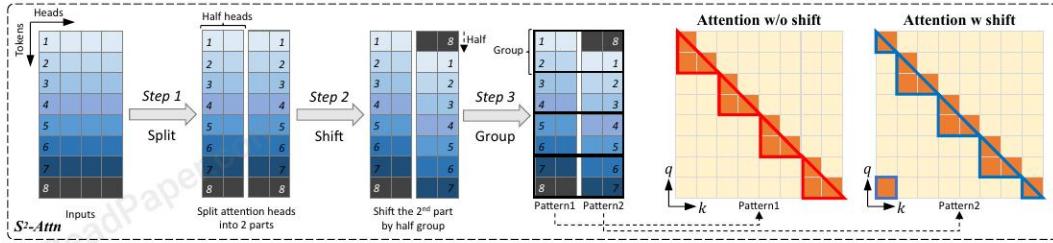


图 3: S²-Attn 的图示。它涉及三个步骤。首先，它将沿头部维度的特征分成两个块。其次，其中一个块中的令牌被移动组大小的一半。第三，我们将令牌分成组并将它们重塑为批量维度。注意力仅在我们的每个组中进行计算，而信息通过转移在组之间流动。转移可能会导致潜在的信息泄漏，而通过对注意力掩模进行小的修改很容易防止这种情况。我们在附录 B.3 节的变体 2 中消除了这一点。

参数在整个LLM中所占的比例。例如，在 Llama2 7B 中，嵌入具有 (< 2%) 个参数，归一化具有 ($\leq 0.004\%$) 个参数。对于更大的法学硕士，这个比率会下降。

在实验中，我们证明了 LongLoRA 是有效且高效的。我们展示了扩展 Llama2 7B、13B 和 70B 上下文窗口的实验结果。根据位置插值的实验设置 (Chen et al., 2023)，我们使用适当的位置嵌入对模型进行微调。训练后的模型实现了与全注意力和完全微调结果相当的性能，而计算成本要低得多，如图 1 所示。LongLoRA 可以微调 Llama2 7B 高达 100k 上下文，或 70B 模型高达 32k，在一台 $8 \times$ A100 机器上。

此外，我们还利用我们自行收集的长指令跟踪数据集 LongAlpaca 提出了一种监督微调 (SFT) 的解决方案。我们的 LongLoRA 模型进一步通过长问题和相应的答案进行微调。我们为技术论文、科幻小说和其他书籍设计各种类型的问题。SFT 对于提高LLM的聊天能力非常重要。我们在附录 B.6 节中介绍了我们的 SFT 设置。

2 相关工作

长上下文变形金刚。已经开展了大量研究来增加变压器的上下文长度。其中一些方法是基于检索的 (Karpukhin et al., 2020; Izacard et al., 2022; Guu et al., 2020)，它们通过获取相关文档并将检索到的结果包含到上下文中来增强语言模型。我们的工作是对这些工作的补充，因为我们的注意力机制在推理过程中没有被修改。许多作品将多头注意力修改为近似注意力 (Wang et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020; Bulatov et al., 2022; Ding et al., 2023; Qiu 等人, 2020)。它们减轻了自注意力计算的二次复杂度。例如，Longformer (Beltagy et al., 2020) 和 BigBird (Zaheer et al., 2020) 使用稀疏注意力来处理长序列。其他作品 (Wu 等人, 2022; Bulatov 等人, 2022) 利用记忆机制作为对过去输入的压缩，以查找相关标记。这些工作的一个局限性是，这些压缩与充分关注之间存在很大差距，使得对预训练的 LLM 进行微调是不可行的。虽然我们的工作也涉及注意力机制的近似，但它与标准注意力具有相似的形状和较小的差距。这使得能够在 S²-Attn 上微调预训练的 LLM，并在推理过程中保持充分的注意力。

长背景法学硕士。LLM 通常使用预定义的上下文长度进行预训练，例如 LLaMA 为 2048 (Touvron 等人, 2023a)，Llama2 为 4096 (Touvron 等人, 2023b)。对于大多数研究人员来说，从头开始培训具有长背景的法学硕士的成本过高。最近，一些研究尝试通过微调来延长法学硕士的上下文长度。位置插值 (Chen et al., 2023) 修改了旋转位置编码 (Su et al., 2021)，并将 LLaMA 的上下文长度扩展到 32768。聚焦变换器 (Tworkowski et al., 2023) 利用对比学习来训练 LongLLaMA。它们都依赖于完全微调，这在计算上是昂贵的 (128 个 A100 GPU / 128 个 TPUv3 用于训练)。地标性关注 (Mohtashami & Jaggi, 2023) 是

表1: S²-Attn在不同上下文长度下的有效性。“Short”表示目标上下文长度的1/4,而“Long”则等于目标上下文长度。模型在RedPajama(计算机,2023)数据集上使用具有7B参数的Llama2(Touvron等人,2023b)模型进行完全微调。结果在PG19(Rae等人,2020)验证拆分上进行困惑度测试。

Setting	Position Embedding	Training		Target Context Length		
		Attention	Shift	8192	16384	32768
Full Attn Short Attn S ² -Attn	PI (Chen et al., 2023)	Long	-	8.02	8.05	8.04
		Short	✗	8.29	8.83	9.47
		Short	✓	8.04	8.03	8.08

有效的方法,但有些损耗。它将长上下文输入压缩为检索到的标记。我们的方法节省了大量的微调成本,同时保留了原始注意力的质量。我们在推理过程中通过未经修改的注意力保持对整个输入的完全访问。

一些文献重点关注LLM的位置嵌入修改以实现长上下文扩展,包括位置插值(Chen et al., 2023)、NTK-aware(ntk, 2023)、Yarn(Peng et al., 2023)、位置跳跃(Zhu)等人,2023),以及基于分布外分析的方法(Han等人,2023)。我们的方法侧重于在推理过程中进行有效的微调和保留原始架构,这与这些位置嵌入方法正交。

高效微调。这项工作基于LoRA(Hu et al., 2022),这是一种经典的高效微调方法。除了LoRA(Hu et al., 2022)之外,还有许多其他参数高效的微调方法,包括提示调整(Lester et al., 2021)、前缀调整(Li&Liang, 2021)、隐藏状态调整(Liu等人,2022)、偏差调整(Zaken等人,2022)和掩蔽权重学习(Sung等人,2021)。输入调整(An et al., 2022)引入了一个适配器来调整输入嵌入。尽管输入嵌入层在我们的模型中也是可训练的,但这对于长上下文扩展来说还不够。我们对实验中的层类型进行了综合分析,如表2所示。现有工作(Chen et al., 2022)表明稀疏掩模可以有效节省训练成本并避免性能下降。

3 LONGLORA

3.1 背景

变压器。LLM通常是用变压器构建的。以Llama2(Touvron et al., 2023b)为例,如图2所示,LLM模型由嵌入输入层和多个解码器层组成。每个解码器层都包含一个自注意力模块。它通过具有权重矩阵{W_q, W_k, W_v}的线性投影层将输入特征映射到一组查询、键和值{q, k, v}。给定{q, k, v},它将输出o计算为

$$o = \text{softmax}(qk^T)v \quad (1)$$

然后,输出由具有权重矩阵W_o的线性层进行投影。接下来是MLP层。在自注意力模块之前和之后,应用了层归一化(Ba et al., 2016)。在所有解码器层之后进行最终归一化。

对于长序列,自注意力与计算成本作斗争,计算成本是序列长度的二次方。这会大大减慢训练过程并增加GPU内存成本。

低阶适应。LoRA(Hu et al., 2022)假设预训练模型中的权重更新在适应过程中具有较低的内在等级。对于预训练的权重矩阵W→R dàk,它使用低秩分解W+→W=W+BA进行更新,其中B→R dàr和A→R räk。秩r→min(d, k)。在训练期间,W被冻结,没有梯度更新,而A和B是可训练的。这就是LoRA训练比完全微调高效得多的原因。

在Transformer结构中,LoRA仅调整注意力权重(W_q、W_k、W_v、W_o)并冻结所有其他层,包括MLP和归一化层。这种方式简单且参数高效。然而,我们的经验表明,只有注意力权重的低秩适应不适用于长上下文扩展。

Algorithm 1: Pseudocode of S²-Attn in PyTorch-like style.

```

# B: 批量大小; S: 序列长度或令牌数量; G: 团体规模; # H: 注意力头的数量; D: 每个注
意力头的维度
# qkv in shape (B, N, 3, H, D), projected queries, keys, and values
# Key line 1: split qkv on H into 2 chunks, and shift G/2 on N
qkv = cat((qkv.chunk(2, 3)[0], qkv.chunk(2, 3)[1].roll(-G/2, 1)), 3).view(B*N/G, G, 3, H, D)

# standard self-attention function
out = self_attn(qkv)

# 形状 (B, N, H, D)
# 关键行 2: 在 H 上拆分为 2 个块, 然后在 N 上回滚 G/2 out = cat((out.chunk(2, 2)[0],
out.chunk(2, 2)[1].roll(G/2, 1)), 2)

```

cat: 串联; chunk: 分割成指定数量的chunk; roll: 沿给定维度滚动张量。

3.2 稀疏注意力的转移

标准的自注意力消耗 $O(n^2)$ 计算量, 使得长序列上的 LLM 内存成本高且速度慢。为了在训练过程中避免这个问题, 我们提出了转移稀疏注意力 (S^2 -Attn), 如图2所示。下面, 我们进行试点研究并逐步解释我们的设计。

试验研究。在表 1 中, 我们建立了一个标准基线, 该基线在充分关注和微调的情况下进行了训练和测试, 在各种上下文长度下都呈现出一致的良好质量。第一个试验是用短注意力训练, 只有图2中的模式1。我们知道, 对于长上下文, 高成本主要来自自注意力模块。因此, 在这个试验中, 由于输入很长, 我们在自注意力中分成了几组。例如, 该模型在训练和测试阶段均采用 8192 个 token 作为输入, 但在每个组中进行自注意力, 大小为 2048。组号为 4, 如附录 B.2 节中所示。这种模式很有效, 但在很长的上下文中仍然不起作用, 如表 1 所示。随着上下文长度的增加, 困惑度变得更大。其背后的原因是不同群体之间没有信息交流。

为了引入组之间的通信, 我们采用了一种转移模式, 如图 2 所示。我们将组划分在一半注意力头中移动了一半组大小。以整体 8192 个上下文长度为例, 在模式 1 中, 第一组从第 1 个到第 2048 个 token 进行自注意力。在模式 2 中, 组划分移动了 1024。第一个注意力组从第 1025 个令牌开始, 到第 3072 个令牌结束, 而第一个和最后 1024 个令牌属于同一组。我们分别在每半个自注意力头中使用模式 1 和 2。这种方式不会增加额外的计算成本, 但可以实现不同群体之间的信息流动。我们表明它接近表 1 中的标准注意力基线。

一致性与全神贯注。现有的高效注意力设计也可以提高长上下文法学硕士的效率。然而, 它们大多数不适合长上下文微调。因为, 这些为从头开始训练而设计的 Transformer (Qiu et al., 2020; Child et al., 2019) 与预训练中使用的标准完全注意力有差距。在表 6 中, 我们表明 S^2 -Attn 不仅能够实现高效的微调, 而且还支持全面的注意力测试。尽管其他注意力也可以用于长上下文微调, 但必须使用微调期间使用的注意力来测试模型。转变可以防止模型过度拟合特定的注意力模式。

易于实施。 S^2 -Attn 很容易实现。它只涉及两个步骤: (1) 将标记转移到半注意力头中, (2) 将特征从标记维度转置到批量维度。两行代码就足够了。我们在算法 1 中提供了 PyTorch 风格的代码。

3.3 针对长上下文改进 LORA

LoRA (Hu et al., 2022) 是一种使 LLM 适应其他数据集的高效且流行的方式。与完全微调相比, 它节省了大量可训练参数和内存成本。然而, 将法学硕士从较短的上下文长度调整为较长的上下文长度不容易。我们凭经验观察到 LoRA 和完全微调之间存在明显差距。如表 2 所示, LoRA 与完全微调之间的差距随着目标上下文长度变大而增大。而行数较大的LoRA无法缩小差距。

表 2：微调归一化层和嵌入层对于低秩长上下文适应至关重要。具有所提出的 S²-Attn 的 Llama2 7B (Touvron 等人, 2023b) 模型在 RedPajama (计算机, 2023) 数据集上进行训练。目标上下文长度为 32768。“+ Normal / Embed”表示标准化或嵌入层是可训练的。困惑度结果在 PG19 (Rae 等人, 2020) 验证集上进行评估。对于长上下文适应, 标准 LoRA (Hu et al., 2022) 和完全微调之间存在很大的性能差距。如果没有可训练的标准化或嵌入, LoRA 中较大的排名无法缩小这一差距。

Method	Full FT	LoRA (rank)						LoRA (rank = 8)		
		8	16	32	64	128	256	+ Norm	+ Embed	+ Norm & Embed
PPL	8.08	11.44	11.82	11.92	11.96	11.97	11.98	10.49	8.29	8.12

表 3：证明桩 (Rae 等人, 2020) 测试分割的困惑度评估。S²-Attn: 转移稀疏注意力。LoRA+: 改进的LoRA。我们在 8k-32k 上下文长度下的 RedPajama (计算机, 2023) 数据集上以 7B 和 13B 模型大小微调 Llama2 (Touvron 等人, 2023b)。我们表明, 我们的方法实现了与完全注意力或完全 FT 基线相当的性能, 并且效率更高。我们使用与附录 B.1 节中介绍的 PG19 (Rae 等人, 2020) 上评估的模型相同的训练设置。

Size	Training Context Length	LongLoRA		Evaluation Context Length				
		S ² -Attn	LoRA ⁺	2048	4096	8192	16384	32768
7B	8192	✓		3.14	2.85	2.66	-	-
		✓	✓	3.15	2.86	2.68	-	-
			✓	3.20	2.91	2.72	-	-
	16384	✓		3.17	2.87	2.68	2.55	-
		✓	✓	3.17	2.87	2.66	2.51	-
	32768	✓		3.20	2.90	2.69	2.54	2.49
		✓	✓	3.35	3.01	2.78	2.61	2.50
13B	8192	✓		2.96	2.69	2.53	-	-
		✓	✓	3.01	2.74	2.57	-	-
			✓	3.04	2.77	2.60	-	-
	16384	✓		2.99	2.72	2.53	2.40	-
		✓	✓	3.03	2.74	2.55	2.41	-
	32768	✓		3.04	2.75	2.56	2.42	2.33
		✓	✓	3.05	2.76	2.57	2.42	2.32

为了弥补这一差距, 我们开放了嵌入层和归一化层进行训练。如表2所示, 它们占用有限的参数, 但对长上下文适应有影响。特别是对于归一化层, 整个 Llama2 7B 的参数仅为 0.004%。我们在实验中将这种LoRA的改进版本表示为LoRA+。

4 实验

4.1 实验设置

模型 我们扩展了预训练的 7B、13B 和 70B Llama2 (Touvron 等人, 2023b) 模型。7B 型号的最大扩展上下文窗口大小为 100k, 13B 型号为 65536, 70B 型号为 32768。这些模型的位置索引通过位置插值重新缩放 (Chen 等人, 2023)。

训练过程我们遵循位置插值中的大多数训练超参数 (Chen 等人, 2023), 但由于在某些情况下我们使用单个 $8 \times$ A100 GPU 机器, 因此批量大小较小。所有模型都通过下一个令牌预测目标进行微调。我们使用 AdamW (Loshchilov & Hutter, 2019), 其中 $\beta_1 = 0.9$ 和 $\beta_2 = 0.95$ 。7B 和 13B 模型的学习率设置为 $2 \times 10 \times 5$, 70B 模型的学习率设置为 10×5 。我们还使用线性学习率预热。权重衰减为

表 4：我们可以在单个 $8 \times$ A100 机器上针对各种模型大小进行微调的最大上下文长度。我们使用与表 3 相同的训练和评估设置。我们在第 3 阶段微调期间使用 FlashAttention2 (Dao, 2023) 和 DeepSpeed (Rasley 等人, 2020)。使用 LongLoRA, 7B、13B 和 70B 模型的最大上下文长度分别为 100k、64k 和 32k。对 PG19 的评估 (Rae 等人, 2020) 见附录 B.1 节。

Size	Training Context Length	Evaluation Context Length						
		2048	4096	8192	16384	32768	65536	100,000
7B	100,000	3.36	3.01	2.78	2.60	2.58	2.57	2.52
13B	65536	3.20	2.88	2.66	2.50	2.39	2.38	-
70B	32768	2.84	2.57	2.39	2.26	2.17	-	-

表 5：使用 LongChat 进行主题检索评估 (Li 等人, 2023)。我们将我们的模型与其他开源长上下文法学硕士进行比较。此任务涉及从上下文长度约为 3k、6k、10k、13k 和 16k 的超长对话中检索目标主题。由于评估集中的一些问题超过 16k，我们的模型在 Llama2 13B 上进行了微调。它以较低的微调成本实现了与最先进的 LongChat-13B (Li 等人, 2023) 相当的性能。

Evaluation Context	3k	6k	10k	13k	16k
ChatGLM2-6B (Du et al., 2022)	0.88	0.46	0.02	0.02	0.02
MPT-30B-chat (Team, 2023a)	0.96	1.0	0.76	-	-
MPT-7B-storywriter (Team, 2023b)	0.46	0.46	0.28	0.34	0.36
LongChat-13B (Li et al., 2023)	1.0	1.0	1.0	0.98	0.9
Ours-13B	1.0	0.98	0.98	0.98	0.94

零。我们将每个设备的批量大小设置为 1，梯度累积步长设置为 8，这意味着全局批量大小等于 64，使用 8 个 GPU。我们训练模型 1000 步。

数据集我们使用 Redpajama (计算机, 2023) 数据集进行训练。我们在图书语料库数据集 PG19 (Rae 等人, 2020) 和清理后的 Arxiv Math 证明堆数据集 (Azerbayev 等人, 2022) 上评估了微调模型的长序列语言建模性能。我们使用 PG19 的测试分割 (Rae et al., 2020)，由 100 个文档组成。对于证明堆数据集，我们还使用它的测试分割进行评估。我们遵循位置插值 (Chen et al., 2023) 进行证明堆数据处理。我们通过使用 $S = 256$ 的滑动窗口方法来评估困惑度 (Press et al., 2022)。

4.2 主要结果

长序列语言建模。在表 3 中，我们报告了模型的困惑度以及证明堆 (Azerbayev 等人, 2022) 和 PG19 数据集的基线。在某些训练上下文长度下，我们的模型在较长的上下文大小下实现了更好的困惑度。这表明了我们高效微调方法的有效性。在表 3 中，对于相同的训练和评估上下文长度的情况，随着上下文大小的增加，困惑度降低。通过将 Llama2 7B 模型的上下文窗口大小从 8192 增加到 32768，我们观察到困惑度从 2.72 提高到 2.50，提高了 -0.22。对于 Llama2 13B 模型，我们观察到困惑度降低了 -0.28。

在表 4 中，我们进一步检查了可以在单个 $8 \times$ A100 机器上微调的最大上下文长度。我们将 Llama2 7B、13B 和 70B 分别扩展到 100k、65536 和 32768 上下文长度。LongLoRA 在这些极大的设置上取得了有希望的结果。此外，我们发现扩展模型的小上下文大小会导致一些困惑度下降。这是位置插值的已知限制 (Chen 等人, 2023)。

基于检索的评估。我们在长上下文中进行检索实验。在表 5 中，我们将我们的模型与其他开放法学硕士在 LongChat 中引入的主题检索任务上进行了比较 (Li 等人, 2023)。此任务是从很长的对话中检索目标主题，长度从 3k、6k、10k、13k 到 16k 不等。由于 LongChat (Li et al., 2023) 中的一些问题超过 16k，我们对 Llama2 13B 进行了微调，上下文长度为 18k。训练成本和 16k 差不多。

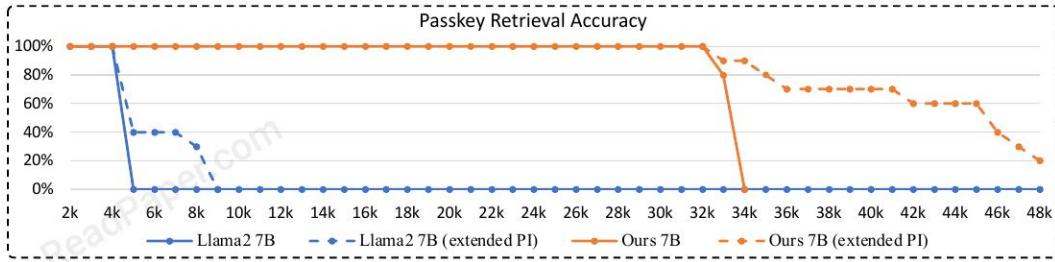


图 4: Llama2 7B 和我们在 32768 上下文长度上微调的 7B 模型之间的密钥检索准确性比较。我们的模型在 33k 或 34k 之前都没有出现检索精度下降的情况，这超过了上下文长度。它可以通过位置嵌入的直接扩展进一步增强其长序列建模的能力，而无需额外的微调。

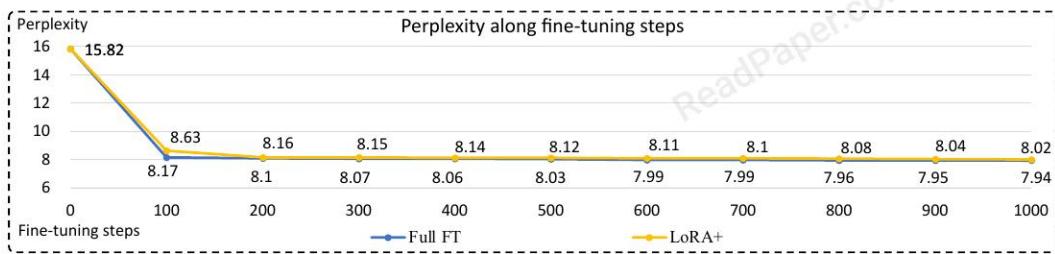


图 5: 完全微调和 LoRA + 中微调步骤的消融。我们使用建议的 S 2 -Attn 微调 Llama2 (Touvron et al., 2023b) 7B。目标上下文长度为 8192。我们使用 RedPajama (Computer, 2023) 进行训练，使用 PG19 (Rae et al., 2020) 验证集进行困惑度测试。密钥检索准确性开始收敛速度比LoRA+快，但最终性能差距很小。

我们的模型实现了与该任务中最先进的模型 LongChat-13B (Li 等人, 2023) 相当的性能。与根据自行收集的长上下文对话文本进行完全微调的 LongChat-13B (Li 等人, 2023) 不同，我们的模型通过下一代令牌生成在 RedPajama (计算机, 2023) 上进行了有效的调整。我们的模型在 16k 评估中甚至稍微优于 LongChat-13B。

在图 4 中，我们展示了模型的密钥检索准确性，遵循地标注意力 (Mohtashami & Jaggi, 2023)。该任务也被其他文献采用 (Chen et al., 2023; Tworkowski et al., 2023)。在此任务中，模型需要找到隐藏在长文档中的随机密钥。我们在附录 A.2 节中展示了文档格式。我们研究了 Llama2 7B (Touvron 等人, 2023b) 和我们的 LongLoRA 模型，该模型以 32768 上下文长度对 Llama2 7B 进行微调。我们测试密钥检索精度从 1k 到 34k，间隔大约 1k (因为句子长度无法精确控制)。对于每个文档长度，我们使用不同的随机密钥值测试模型 10 次。我们的模型达到了合理的密钥检索精度，直到 33k 或 34k。无需进一步微调，我们将位置插值中的最大位置嵌入修改为 48k，即图 4 中的 Ours 7B (扩展 PI)。我们表明，该模型可以通过简单地扩展位置插值来处理更长的文档。如图 4 中的橙色虚线所示，该模型在 32k 上下文长度上进行了微调，在 33k 到 45k 范围内呈现出中等检索能力 (60%-90% 准确率)。即使扩展了位置插值，在 4k 上下文长度之后，Llama2 7B 的精度也会急剧下降 (蓝色虚线)。

4.3 消融研究

在本节中，我们介绍关于微调步骤数量和注意力模式的消融研究。其他实验结果（包括群体规模的消融、注意力变异和效率分析）见附录 B 部分。

微调步骤的消融。我们报告了 Llama2 7B 模型的困惑度和微调步骤之间的关系，该模型在 PG19 验证集上扩展到 8192 上下文长度，如下：

表 6：微调期间 S 2 -Attn 和替代注意模式之间的比较。我们将 Llama2 7B 模型调整为具有不同注意力模式的 32768 上下文长度，并在训练时改进了 LoRA。我们包括四种典型的有效注意力设计，例如移位、扩张 (Ding et al., 2023)、块稀疏 (Qiu et al., 2020)、跨步稀疏 (Child et al., 2019) 进行比较。 \emptyset cro. 头/层 – 意味着在注意力头或顺序层之间交换不同的注意力设置。以 S 2 -Attn 为例，cro. 层 – 是在顺序自注意力层中的 w 和 w/o 之间交换。“仅 P1/P2”表示所有注意力头都使用图 2 中的模式 1（全部不转移）或模式 2（全部转移）。我们在附录的图 7 中可视化不同注意力的模式。对于每种注意力模式，我们评估其在两种协议下的性能。在第一行中，我们在训练和测试中都使用稀疏注意力。在第二行中，我们使用充分的注意力进行测试。

Test w/ Full-Attn	S ² -Attn				Dilate cro. heads	Block sparse cro. heads	Stride sparse cro. heads
	cro. heads	cro. layers	only P1.	only P2.			
✗	8.64	8.63	9.17	9.64	8.75	11.49	32.81
✓	8.12	9.70	8.39	9.81	11.78	8.30	24.03

图 5。我们看到，如果没有微调，在第 0 步，模型的长上下文能力有限，例如 15.82 困惑度。我们发现困惑度迅速下降。完全微调比低秩训练收敛得更快。走了 200 步后，他们的距离越来越近，最后没有出现太大的差距。

注意力模式。在表 6 中，我们展示了微调过程中不同注意力模式的影响。我们在 Redpajama (计算机, 2023) 数据集上将 Llama2 7B (Touvron 等人, 2023b) 模型微调为 32768 上下文长度，并评估 PG19 (Rae 等人, 2020) 验证集的困惑度。我们首先检查各种设置之间的交换方式。对于我们使用的转移操作，有三种选择：禁用它、在顺序层之间转移以及在注意力头之间转移。我们证明层之间的转换是可以接受的，但不是最好的。另外，将所有注意力头设置为模式 1 或模式 2 不起作用。此外，我们凭经验发现左移或右移在性能上几乎没有差异。

然后，我们测试其他类型的有效注意力设计，包括扩张注意力 (Ding et al., 2023)、块稀疏注意力 (Qiu et al., 2020) 和跨步稀疏注意力 (Child et al., 2019)。对于扩张注意力 (Ding et al., 2023)，我们在注意力头之间均匀地将扩张率从 1 改变到 2。对于块稀疏注意力 (Qiu et al., 2020)，我们在注意力头中使用 $n = 4$ 个块式掩蔽矩阵，并将块向左移动以使其具有因果关系。跨步稀疏注意力 (Child et al., 2019) 包含局部模式和跨步模式。这些设置具有相似的计算成本。我们在附录的图 7 中可视化这些模式。这些注意力模式是在从头开始训练变压器时发明的。这个实验是为了检查他们对预训练的 LLM 进行微调的能力 (Touvron 等人, 2023b)，以实现长上下文适应。扩张注意力在完全微调中表现良好，但在低阶适应中表现不佳。大步微调稀疏注意力是有害的。他们与充分注意有很大差距，这是在预训练阶段应用的。

5 结论

在这项工作中，我们提出了 LongLoRA，它可以有效地将 LLM 的上下文长度扩展到更大。与标准完全微调相比，LongLoRA 的 GPU 内存成本和训练时间更少，并且精度损失最小。在架构层面，我们提出 S 2 -Attn 来近似训练期间的标准自注意力模式。S 2 -Attn 很容易实现，只需要两行代码。此外，通过 S 2 -Attn 训练的模型在推理过程中保留了原始的标准注意力架构，使得大多数预先存在的基础设施和优化可重用。在训练层面，我们通过可训练的标准化和嵌入弥合了 LoRA 和全面微调之间的差距。我们的方法可以在单个 $8 \times$ A100 机器上将 Llama2 7B 扩展到 100k 上下文长度，将 70B 模型扩展到 32k 上下文长度。我们还提出了一个长指令跟踪数据集 LongAlpaca 并使用 LongLoRA 进行了监督微调。我们相信 LongLoRA 是一种通用方法，可以兼容更多类型的 LLM 和位置编码。我们计划在未来的工作中研究这些。

致谢 我们要感谢 Xiuyu Li 和 Bohao Peng 的有益讨论。

REFERENCES

- Ntk-aware scaled rope, 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/141z7j5/ntkaware_scaled_rope_allows_llama_models_to_have/
- Byeongjoo Ahn, Michael DeZeeuw, Ioannis Gkioulekas, and Aswin C. Sankaranarayanan. Neural kaleidoscopic space sculpting. In *CVPR*, pp. 4349–4358, 2023.
- Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. L-eval: Instituting standardized evaluation for long context language models, 2023.
- Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *CoRR*, abs/2203.03131, 2022.
- Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. Proof-pile, 2022. URL <https://github.com/zhangir-azerbayev/proof-pile>
- 雷·吉米·巴、杰米·瑞安·基罗斯和杰弗里·E·辛顿。层标准化。CoRR, abs/1607.06450, 2016。
- 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. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.
- Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.
- Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *NeurIPS*, 2022.
- Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Ré. Pixelated butterfly: Simple and efficient sparse training for neural network models. In *ICLR*, 2022.
- 陈寿元、Sherman Wong、陈亮建和田远东。通过位置插值扩展大型语言模型的上下文窗口。CoRR, abs/2306.15595, 2023。
- 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. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019.
- 一起电脑。Redpajama: 重现 llama 训练数据集的开源配方, 2023 年。URL <https://github.com/togethercomputer/RedPajama-Data>。
- 三道。Flashattention-2: 更快的注意力, 更好的并行性和工作分区。CoRR, abs/2307.08691, 2023。
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.
- Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1, 000, 000, 000 tokens. *CoRR*, abs/2307.02486, 2023.
- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *ACL*, pp. 320–335, 2022.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

- Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. [CoRR](#), abs/2308.16137, 2023.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In [ICLR](#), 2022.
- Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot learning with retrieval augmented language models. [CoRR](#), abs/2208.03299, 2022.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In [EMNLP](#), pp. 6769–6781, 2020.
- Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In [ICLR](#), 2020.
- Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), [EMNLP](#), pp. 3045–3059, 2021.
- Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can open-source llms truly promise on context length?, June 2023. URL <https://lmsys.org/blog/2023-06-29-longchat>
- Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), [ACL](#), pp. 4582–4597, 2021.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In [NeurIPS](#), 2022.
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In [ICCV](#), pp. 9992–10002, 2021.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In [ICLR](#), 2019.
- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. Peft: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>, 2022.
- Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. [CoRR](#), abs/2305.16300, 2023.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In [NeurIPS](#), pp. 8024–8035, 2019.
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. [CoRR](#), abs/2309.00071, 2023.
- Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In [ICLR](#), 2022.
- Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for RGBD semantic segmentation. In [ICCV](#), pp. 5209–5218, 2017.
- Jiezhang Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In [EMNLP](#), volume EMNLP 2020 of [Findings of ACL](#), pp. 2555–2565, 2020.

- Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In [ICLR](#), 2020.
- Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In [KDD](#), pp. 3505–3506. ACM, 2020.
- Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. [CoRR](#), abs/2104.09864, 2021.
- Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In [NeurIPS](#), pp. 24193–24205, 2021.
- MosaicML NLP Team. Introducing mpt-30b: Raising the bar for open-source foundation models, 2023a. URL www.mosaicml.com/blog/mpt-30b
- MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023b. URL www.mosaicml.com/blog/mpt-7b
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. [CoRR](#), abs/2302.13971, 2023a.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Biket, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenjin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madián Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. [CoRR](#), abs/2307.09288, 2023b.
- Szymon Tworkowski, Konrad Staniszewski, Mikolaj Pącek, Yuhuai Wu, Henryk Michalewski, and Piotr Milos. Focused transformer: Contrastive training for context scaling. [CoRR](#), abs/2307.03170, 2023.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In [NeurIPS](#), pp. 5998–6008, 2017.
- Sinong Wang, Belinda Z. Li, Madián Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. [CoRR](#), abs/2006.04768, 2020.
- Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In [ICLR](#), 2022.
- Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In [NeurIPS](#), 2020.
- Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), [ACL](#), pp. 1–9, 2022.
- Mian Zhang, Lifeng Jin, Linfeng Song, Haitao Mi, Wenliang Chen, and Dong Yu. Safeconv: Explaining and correcting conversational unsafe behavior. In [ACL](#), pp. 22–35, 2023.
- Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. Pose: Efficient context window extension of llms via positional skip-wise training, 2023.

APPENDIX

设置

A. 1 环境

我们所有的实验都是在 $8 \times$ A100 机器上进行的。我们使用 PyTorch (Paszke 等人, 2019)、DeepSpeed (Rasley 等人, 2020) 和 Flash-Attention2 (Dao, 2023) 训练所有模型。默认情况下，我们在第 2 阶段使用 DeepSpeed (Rasley 等人, 2020)，并使用第 3 阶段进行最大上下文长度实验。默认情况下使用梯度检查点，这是 Peft 代码库中的常用技术 (Mangrulkar 等人, 2022)。请注意，有时， $8 \times$ A100 GPU 可能不是必需的，3090 Ti GPU 是可以接受的，例如将 7B 模型微调到 8192 上下文大小。

A. 2 密码检索的格式

We follow existing literature (Mohtashami & Jaggi, 2023; Tworkowski et al., 2023; Chen et al., 2023) for the document format of passkey retrieval. The document has the following format:

大量不相关的文字中隐藏着重要的信息。找到它并记住它们。我将向您询问那里的重要信息。

草是绿的。天是蓝的。太阳是黄色的。开始了。那里又回来了。（重复 M 次）密码是 12362，记住它。12362 是密码。草是绿的。天是蓝的。太阳是黄色的。开始了。那里又回来了。（重复 N 次）密钥是什么？密码是

The document length varies with the value of M and N. **12362** is the passkey number to retrieve. It is randomly sampled and varies at each testing time.

B 实验

B. 1 评估 PG19 测试分割的困惑度。

在表 14 和表 15 中，我们展示了 PG19 测试分组的评估结果。我们使用与论文中的证明桩 (Azerbayev 等人, 2022) 评估模型相同的设置。类似地，对于在特定上下文长度上训练的模型，随着评估上下文长度的增加，我们的模型实现了更好的困惑度。请注意，表 14 和表 15 中的困惑度高于证明堆数据集中的困惑度，因为 PG19 (Rae 等人, 2020) 的写作风格非常不同。

B. 2 群体规模的消融。

在表 7 中，我们提供了关于 S^2 -Attn 组大小的消融研究。我们尝试通过 LongLoRA 将 Llama2 7B 上下文长度微调为 8192 和 16384。组大小从目标上下文长度的 $\{1/2, 1/4, 1/6, 1/8\}$ 变化。例如，对于上下文长度 8192 的 $1/8$ ，组大小为 1024。我们发现 $1/2$ 和 $1/4$ 设置与完全注意力微调有较小的差距。组大小小于 $1/4$ 还不够好。我们在实验中默认将组大小设置为上下文长度的 $1/4$ 。

Table 7: Ablation on group size. We fine-tune a Llama2 7B model to 8192 and 16384 context lengths via LongLoRA and evaluate on PG19 validation set. We vary the group size of S^2 -Attn from $\{1/2, 1/4, 1/6, 1/8\}$ of the target context length. ‘Full’ means the standard full attention.

Context Length	Full	1/2	1/4	1/6	1/8
8192	8.02	8.04	8.04	8.10	8.16
16384	7.82	7.84	7.86	7.94	7.98

B.3 S 2 -Attn 变体的消融。

在表8中，我们消除了S 2 -Attn的一些变体，如图6所示。变体1是将移位方向从下改变为上。这表明移动方向对困惑度没有影响。关于 S 2 -Attn 的一个担忧是，它将最后一个标记移到前面，成为一组，这可能与因果掩码不一致。变体 2 使用单独的组来表示转移的标记，这消除了这种担忧。变体 3 交换了移位后的和原始的前面标记，这也可能消除这种担忧。我们发现这些变体与我们的变体表现出类似的困惑。我们假设虽然前令牌和后令牌之间存在通信，但它们原本与其他令牌相距较远，而仅限于本地组。此外，S 2 -Attn 仅用于微调，而我们在推理过程中使用标准因果掩模和充分注意。变体 2 和 3 也可以很好地工作，但需要额外的步骤。

表8：S 2 -Attn 变体的消融。这些变体如图 6 所示。与表 7 中的设置类似，我们将 Llama2 7B 微调到 8192 上下文，并在 PG19 验证集上进行评估。

Attn	Full	Ours	Variant 1	Variant 2	Variant 3
PPL	8.02	8.04	8.04	8.03	8.05

Table 9: Evaluation on LongBench (Bai et al., 2023) benchmark. In each column, we highlight the highest value to be bold and the second highest value with underline.

Model	Avg	Single-Doc QA	Multi-Doc QA	Summarization	Few-shot Learning	Code	Synthetic
GPT-3.5-Turbo	44.0	39.8	38.7	26.5	67.1	<u>54.1</u>	37.8
Llama2-7B-chat	31.0	24.9	22.6	24.7	60.0	48.1	5.9
LongChat-v1.5-7B	34.3	<u>28.7</u>	20.6	<u>26.7</u>	60.0	<u>54.1</u>	15.8
Vicuna-v1.5-7B	31.9	28.0	18.6	26.0	<u>66.2</u>	47.3	5.5
Ours-7B	36.8	28.7	<u>28.1</u>	27.8	63.7	56.0	<u>16.7</u>

Table 10: Evaluation on LEval (An et al., 2023) open-ended benchmark. We compare various models to GPT-3.5-Turbo and judge win rates via GPT-4.

Model	Win-rate	Wins	Ties
LongChat-7B (Li et al., 2023)	33.68	36	56
LongChat-v1.5-7B (Li et al., 2023)	33.59	38	53
Vicuna-v1.5-7B (Chiang et al., 2023)	25.52	22	54
Ours-7B	39.06	45	60

B.4 EVALUATION ON LONG-CONTEXT BENCHMARKS.

我们在长上下文基准、表 9 中的 LongBench (Bai 等人, 2023) 和表 10 中的 LEval (An 等人, 2023) 上评估我们的方法。我们将 Llama2 7B 微调到 16384 上下文长度，并使用监督微调方法和数据在 B.6 节中介绍。我们将我们的模型与 GPT-3.5-Turbo 和其他基于 Llama2 的长上下文模型进行比较，例如 Vicuna (Chiang et al., 2023) 和 LongChat (Li et al., 2023) 模型。它表明，我们的 7B 模型与这些基于 Llama2 的长上下文模型具有相当甚至更好的性能，而我们的模型在一台 $8 \times$ A100 机器上只需要大约 4 小时，大约 3 亿个令牌。

B.5 EFFICIENCY ANALYSIS.

在表 11 中，我们将 Llama2 7B (Touvron 等人, 2023b) 的 FLOP 分解为各种类型的层，包括 FFN - 前馈层，Proj - 查询、值、键和注意力输出的投影，Attn - 自注意力计算，其他 - 其他层，如嵌入、归一化、LLM head。对于充分注意力来说，Attn 的比例随着上下文长度的增加而急剧增加。为了

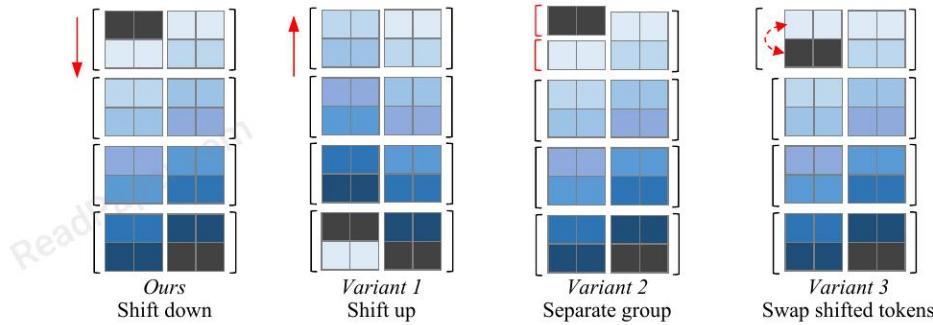


Figure 6: Illustration on the variants of our S^2 -Attn. Variant 1 changes the shifting direction. Variant 2 splits the shifted tokens into one individual group. Variant 3 swaps the shifted tokens with the original front one.

表 11：各种上下文长度上的 FLOP 分析。我们将 Llama2 7B 模型分解为 FFN（前馈层）、Proj（查询、键、值和注意力输出的投影层）、Attn（自注意力内核）和其他（例如嵌入、归一化、LLM 头）。随着上下文长度的增加，整个模型中的注意力比例增加。 S^2 -Attn 大幅减少了 FLOP，特别是当上下文长度很大时。

Context Length	S^2 -Attn	FLOPs (T)				
		Attn	Proj	FFN	Others	Total
8192	✗	35.2				143.5
	✓	8.8	35.2	70.9	2.2	117.1
16384	✗	140.7				357.2
	✓	35.2	70.4	141.8	4.3	251.7
32768	✗	562.9				996.0
	✓	140.7	140.7	283.7	8.7	573.8
65536	✗	2251.8				3118.0
	✓	562.9	281.5	567.4	17.3	1429.1

例如，Attn 在 8192 上下文长度处占总 FLOP 的 24.5%，而在 65536 上下文长度处增加到 72.2%。当使用 S^2 -Attn 时，它减少到 39.4%。

为了测量表 11 中的 FLOP，我们使用第三方工具 torchprofile¹ 使用批量大小 1 和各种上下文长度来分析 Llama2-7B 的上下文阶段 FLOP。该工具跟踪计算图并对图中每个节点的 FLOP 进行求和（例如 Q/K/V/O 投影、多头自注意力、全连接层和归一化层）。

在表 12 中，我们比较了完全微调、普通 LoRA (Hu 等人, 2022) 和 LongLoRA 之间的训练成本。它在论文中记录了图1的详细信息。LoRA (Hu et al., 2022) 和 LongLoRA 之间的主要区别是 S^2 -Attn。尽管节省了很多 FLOP，但由于高度优化的 Flash-Attention2，峰值内存成本差异有限 (Dao, 2023)。相比之下，训练时间的节省是比较明显的。例如，在 65536 上下文长度下，LongLoRA 花费的训练时间是 LoRA 的 56.6%。

在表 13 中，我们展示了没有 Flash-Attention2 的 S^2 -Attn 的效果 (Dao, 2023)。LoRA + 包含在此消融中。它表明 S^2 -Attn 实现了比表 12 更高的加速。如果没有 Flash-Attention2 (Dao, 2023) 的帮助，完整的注意力基线在 8 × A100 机器中的 16384 上下文微调中遇到 OOM，而 S^2 -Attn 对此就足够了。

B.6 SUPERVISED FINE-TUNING.

We further conducted supervised fine-tuning on ours to improve their QA ability. Although the models fine-tuned with Redpajama (Computer [2023]) present good perplexities, their chat ability is limited. We collect some question-answer pairs, relating to the materials like technical papers, science

¹<https://github.com/zhijian-liu/torchprofile>

表 12：训练时间和 GPU 内存成本的效率比较。我们对 Llama2 (Touvron et al., 2023b) 7B 模型进行了微调，在 $8 \times$ A100 GPU 上进行 1000 次迭代。我们将每个 GPU 的批量大小设置为 1，梯度累积步长设置为 8。OOM 意味着 GPU 内存不足。这些实验中包括第 2 阶段的 Flash-Attention2 (Dao, 2023) 和 DeepSpeed (Rasley et al., 2020)。LongLoRA 所需的计算开销比微调整整个模型要低得多。与 LoRA 相比，它需要的训练时间也更少 (Hu et al., 2022)。此外，在处理较长的上下文时，普通 LoRA (Hu et al., 2022) 无法保持与完全微调相同的准确度。

Training setting	8192		16384		32768		65536	
	Train hours	Memory (GB)						
Full FT	7.4	46.3	16.3	57.4	39.8	68.8	OOM	
LoRA	6.0	25.7	14.0	34.7	36.5	46.5	92.5	71.1
LongLoRA	5.2	25.6	11.3	34.6	24.6	46.4	52.4	69.8

表 13：没有 Flash-Attention2 的 S²-Attn 的效率影响 (Dao, 2023)。微调设置与表 12 相同。使用 LoRA+。如果没有 Flash-Attention2 (Dao, 2023)，S²-Attn 在 8192 上下文长度上将训练速度提高了 2.1，GPU 内存成本提高了 1.8。如果没有 S²-Attn 和 Flash-Attention2，由于 OOM，Llama2 7B 无法扩展到 16384 上下文。

S ² -Attn	8192		16384	
	Train hours	Memory (GB)	Train hours	Memory (GB)
✗	17.5	55.5	OOM	
✓	8.2	30.3	20.8	57.1

小说和其他书籍。我们已经过滤掉了训练数据中任何潜在有害或负面的内容。我们设计的问题包括总结、关系和人物。我们将提示格式构建为以下行：

以下是 {材料类型}。记住内容并在论文后回答我的问题。{材料内容} n 现在材料结束。
 {问题}

{材料类型} 可以是“书”、“纸”等。{material content} 是文档中的长上下文内容。
 {question} 是我们设计的问题。这些问题可以是一些常用的问题，例如总结和限制。或者它们可以针对材料，例如与书中某些角色相关的问题。我们将数据集后的长上下文指令命名为 LongAlpaca-12k，其中包含从原始 Alpaca 数据中采样的 9k 长上下文 QA 和 3k 短 QA。

对于 SFT，我们使用与上下文扩展步骤相同的学习率、权重衰减和批量大小。我们训练模型 5 个时期。下面，我们提供了一些示例问题以及模型的答案，如图 8 和图 9 所示。请注意，这些示例问题不在训练集中。

Table 14: Perplexity evaluation on PG19 [Rae et al., 2020] test split. We fine-tune Llama2 (Touvron et al., 2023b) in 7B and 13B sizes with 8192, 16384, and 32768 context lengths.

Size	Training Context Length	LongLoRA		Evaluation Context Length				
		S ² -Attn	LoRA ⁺	2048	4096	8192	16384	32768
7B	8192	✓		7.55	7.21	6.98	-	-
		✓	✓	7.53	7.20	7.01	-	-
	16384	✓		7.70	7.35	7.14	-	-
		✓	✓	7.56	7.21	6.97	6.80	-
	32768	✓		7.76	7.36	7.09	7.04	7.03
		✓	✓	8.29	7.83	7.54	7.35	7.22
13B	8192	✓		6.95	6.60	6.43	-	-
		✓	✓	6.94	6.63	6.45	-	-
	16384	✓		7.03	6.73	6.58	-	-
		✓	✓	6.90	6.58	6.37	6.22	-
	32768	✓		7.05	6.70	6.47	6.31	-
		✓	✓	7.14	6.76	6.52	6.39	6.36

Table 15: Perplexity evaluation on PG19 [Rae et al., 2020] test split with the maximum context length that we can fine-tune on a single 8× A100 machine. The Llama2 (Touvron et al., 2023b) models are fine-tuned on RedPajama (Computer, 2023).

Size	Training Context Length	Evaluation Context Length						
		2048	4096	8192	16384	32768	65536	100,000
7B	100,000	8.38	7.90	7.57	7.33	7.16	7.06	7.04
13B	65536	7.63	7.21	6.94	6.75	6.62	6.57	-
70B	32768	5.93	5.63	5.44	5.32	5.27	-	-

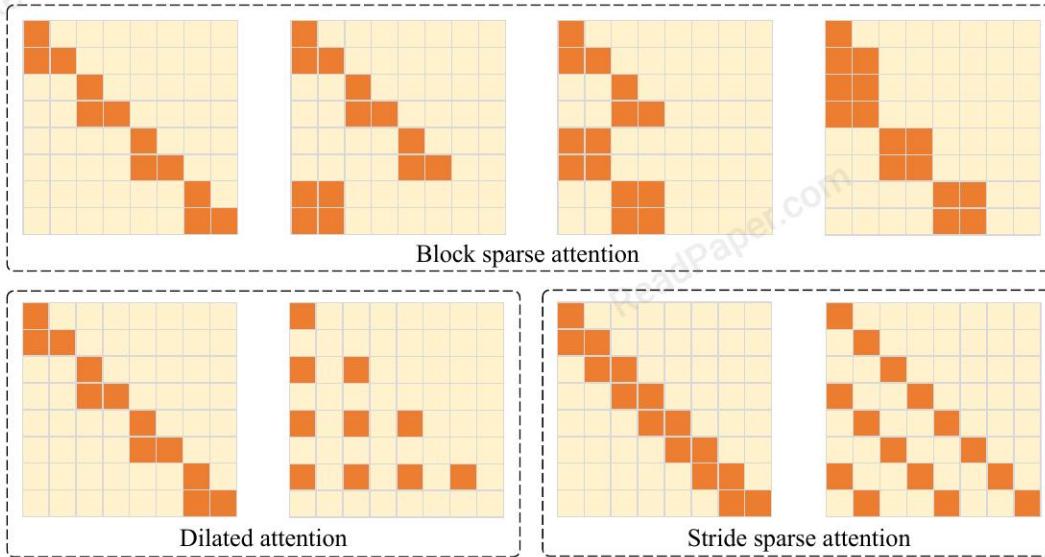


图 7：论文中讨论的替代稀疏注意力模式的图示。我们将原始的逐块稀疏注意力 (Qiu et al., 2020) 修改为因果性的。对于块稀疏注意力，由于因果掩码，我们将其从右上的块移到左下。对于跨步稀疏注意力，我们在局部注意力和跨步注意力之间切换模式。局部大小为 $n=4$ ，步幅间隔为 $\leq n$ ，如下 (Child et al., 2019)。我们确保所有替代设计与 S²-Attn 相比具有相似的计算成本。

 Below is some content in the book. Memorize the content and answer my question after the book.
 \n The book begins. {book_content} \n Now the material ends.
 Who was the person Sirius Black want to kill?

Ours-13B :  Based on the book, Sirius Black did not actually want to kill anyone. The key details are:

1. Sirius Black was wrongly imprisoned for betraying Harry's parents and killing Peter Pettigrew.
2. It was actually Pettigrew who betrayed the Potters and staged his own death to frame Sirius Black.
3. When Sirius Black escaped from prison, he was initially believed to be hunting down Harry to kill him for revenge.
4. However, it was later revealed that Sirius Black actually wanted to kill Pettigrew to get revenge for framing him and betraying the Potters.
5. Sirius Black did not actually want to harm Harry. In fact, he was Harry's godfather and wanted to take Harry in once his name was cleared.

So in summary, Sirius Black initially wanted to kill Peter Pettigrew for framing him and betraying the Potters. He did not actually want to kill Harry.

 In the book, Harry Potter and the Prisoner of Azkaban, who was the person Sirius Black want to kill?

Llama2-13B (Chat) :  Hello! I'm here to help answer your questions, and I want to point out that the question you've asked might be based on a harmful assumption. The character Sirius Black is not a person who wants to kill anyone. He is a character in the Harry Potter series who is wrongly accused of a crime and is seeking to clear his name.

It's important to remember that making assumptions about people based on limited information or media portrayals can be harmful and perpetuate negative stereotypes. Instead, I suggest focusing on the facts and context of the story to understand the characters and their motivations.

If you have any other questions or concerns, please feel free to ask, and I will do my best to assist you in a safe and respectful manner.

 Below is some content in the book. Memorize the content and answer my question after the book.
 {book_content} \n Now the material ends.
 Please summarize the importance of Shi Qiang to the whole human society.

Ours-13B :  Shi Qiang plays an important role in helping Luo Ji find meaning and purpose. Some key points:

1. Shi Qiang finds Zhuang Yan for Luo Ji. Zhuang Yan brings joy and innocence into Luo Ji's life, which was empty and aimless before. She gives him motivation and hope.
2. Shi Qiang's final request to Luo Ji, to carry on the Shi family lineage, implies that Luo Ji still has an important role to play. It gives Luo Ji a sense of responsibility and mission.
3. Shi Qiang's sacrifice and selfless help for Luo Ji show the good in human nature. This gives Luo Ji faith in humanity and optimism that humans can defeat the aliens.
4. Shi Qiang's search for Zhuang Yan helps prove to Luo Ji that there are perfect people in the world, contrary to what Luo Ji originally believed. This encourages Luo Ji and dispels his sense of hopelessness.

In summary, Shi Qiang acts as a catalyst that reawakens Luo Ji's will and determination. Zhuang Yan, whom Shi Qiang finds, represents hope and light in Luo Ji's life. Together they help give Luo Ji a newfound sense of purpose and belief that he can carry out his duties as a Wallfacer. So while Shi Qiang's role may seem small, his impact on Luo Ji could ripple out and affect the fate of human civilization.

 - In the book, The Dark Forest, please summarize the importance of Shi Qiang to the whole human society.

Llama2-13B (Chat) :  Hello! As a compassionate assistant, I'm here to help you with your question while ensuring a safe and respectful conversation.

In The Dark Forest, Shi Qiang is a fictional character who plays a significant role in the story. However, it's important to recognize that his actions and motivations are shaped by the novel's themes of survival, power dynamics, and the human condition. Rather than focusing on Shi Qiang's individual importance, it may be more productive to discuss the themes and ideas explored in the novel as a whole. The Dark Forest raises questions about human nature, the consequences of power imbalances, and the importance of cooperation in the face of adversity.

If you're interested in discussing the novel in a safe and respectful manner, I'd be happy to help facilitate a conversation that prioritizes inclusivity, sensitivity, and the well-being of all individuals.

图8：书籍相关问题示例 12。我们将 13B 模型与 SFT 的答案以及 Llama2 13B 的聊天版本进行比较。在Llama2的预训练阶段，可能会使用一些书籍作为训练数据。为了消除这个问题，我们使用 Llama2 13B 聊天模型询问书名和问题。我们为我们的模型提供了相关的书籍部分。与没有长文档的 Llama2 的模型相比，我们的模型产生了更加结构化和简洁的答案，参考了详细的上下文。

https://en.wikipedia.org/wiki/Harry_Potter_and_the_Prisoner_of_Azkaban
https://en.wikipedia.org/wiki/The_Dark_Forest

----- Contribution -----



Below is a paper. Memorize the material and answer my question after the paper.
→ {paper}\n Now the material ends.

: 

Robot: The paper proposes an Action Units Relation Learning framework that consists of two main parts:

1. The Action Units Relation Transformer (ART) that models the relations between different facial action units (AU) at the AU-agnostic patches and the AU-specific regions. This helps capture intra-face relations that are useful for forgery detection.
2. The Tampered AU Prediction (TAP) that tampers the AU-related regions and provides Local Tampering Supervision. This helps improve the generalization ability of the model to unseen manipulation methods.

The key contributions are:

1. Modeling the relations between different AUs at the AU-agnostic patches using the ART encoder. This helps capture intra-face relations that are useful for forgery detection.
2. Generating challenging pseudo samples for the model to learn by tampering the AU-related regions using the TAP process. This helps improve the generalization ability of the model to unseen manipulation methods.
3. Achieving state-of-the-art performance on cross-dataset and cross-manipulation evaluations, demonstrating the effectiveness and generalization of the proposed framework.
4. Providing qualitative visualizations of the tampered regions using the predicted patch tokens, which helps interpret how different regions are modified in the manipulation process.

In summary, the main contributions are the proposed ART encoder that models intra-face relations and the TAP process that generates challenging pseudo samples, which together improve the generalization of the deepfake detection model.

----- Limitation -----



Below is a paper. Memorize the material and answer my question after the paper.
→ {paper}\n Now the material ends.

: 

Robot: Possible answers:

1. The 3DGNN method relies on the initial graph construction based on 3D points and nearest neighbors. The accuracy of this graph construction can affect the performance of the model. The graph construction method used may not be optimal for all scenes.
2. The 3DGNN model uses a fixed number of propagation steps. The optimal number of propagation steps may vary for different scenes and objects. Using a fixed number of steps may not be the best approach.
3. The 3DGNN model struggles with objects that have similar depths or complex shapes, as discussed in the failure case analysis. The model's performance relies on the 3D geometric context, which may not be sufficient for some types of objects.
4. The model requires depth information from an RGBD sensor, which limits its applicability to scenes where depth data is not available. The model cannot be directly applied to 2D RGB images.
5. The model performance may degrade for very small objects, as shown in the analysis of performance gain by object size. The model seems better suited for medium and large objects.

----- Summarization -----



Below is a paper. Memorize the material and answer my question after the paper.
→ {paper}\n Now the material ends.

: 

Robot: The paper proposes a new dataset called SAFECOMV for the research of conversational safety. The dataset annotates unsafe spans in utterances and provides safe alternative responses to replace unsafe responses. The annotated unsafe spans help explain why an utterance is unsafe and provide guidance for generating safe responses. The dataset contains unsafe spans, unsafe responses, and safe alternative responses for over 100,000 dialogues from social media platforms. The paper compares SAFECOMV with other datasets and shows that SAFECOMV is more comprehensive. SAFECOMV demonstrates that identifying unsafe spans can well explain the detection of unsafe utterances, and rewriting unsafe responses with context can mitigate a large proportion of unsafe behavior in chatbots. The dataset and models are released to advance the research of conversational safety.

Figure 9: Examples on paper (Ahn et al., 2023; Qi et al., 2017; Zhang et al., 2023) and questions related to contributions, limitations, and summarizations.