LongLoRA

https://readpaper.com/paper/4802801264781950977

[PDF] LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models-论文阅读讨论-ReadPaper - 轻松读论文

We present LongLoRA, an efficient fine-tuning approach that extends the conte···

https://github.com/dvlab-research/LongLoRA

GitHub - dvlab-research/LongLoRA: Code and documents of LongLoRA and LongAlpaca (ICLR 2024 Oral)

Code and documents of LongLoRA and LongAlpaca (ICLR 2024 Oral) - dvlab-···

摘要

本文提出了LongLoRA，一种在拓展LLM上下文窗口的高效微调方法。通常，训练一个长上下文窗口的LLM计算成本很高，从2K增长到8K就需要增加16倍的计算成本。本文从两个方面提升了LLM的上下文窗口。

1. 虽然在推理阶段需要使用dense global attention，但是可以使用sparse local attention来对模型进行微调。本文提出的shifted sparse attention在sparse local attention基础上有效地实现了对上下文的拓展，减少了计算资源损耗，并取得了与全量微调相近的效果。需要特别指出的是：这种方法可以在训练中仅使用两行代码实现，并在推理过程中可选。
2. 本文重新审视了PEFT的机制，发现将用于上下文拓展的LoRA应用于trainable embedding and normalization时能取得很好的效果。

LongLoRA在拓展上下文的同时能保留LLM的原始结构，因此与现有的许多方法（如Flash-Attention）适配。

论文试图解决什么问题

在微调层面如何提升LLM上下文长度。

首先想到的方法自然是微调，但是全量微调方法效果昂贵，PI使用了 32 A100 将Llama从2k拓展到了8k， 128 A100用于更长的上下文微调。

另外一个想法便是使用LoRA微调。但是本文的实验表明直接使用LoRA 微调neither sufficiently effective nor efficient。

effective：增大rank对于PPL没有明显缓解。

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

Llama2 7B target context length is 32768

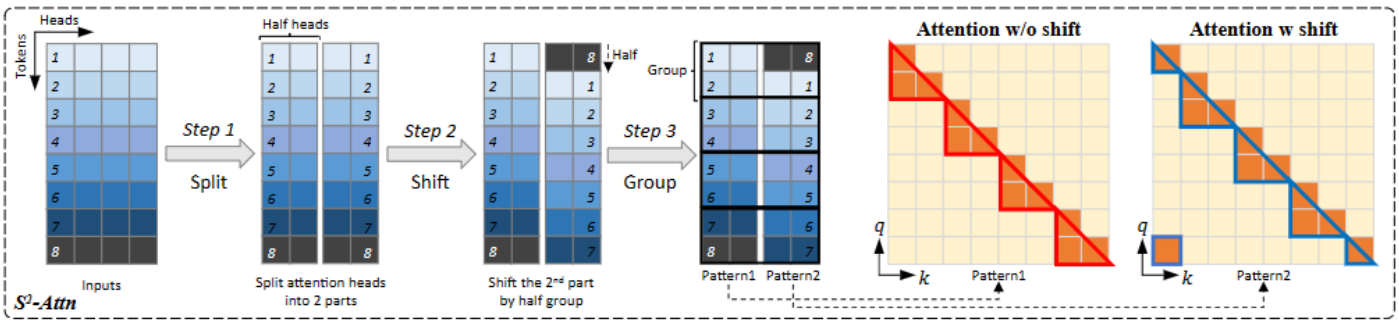
efficient：无论使用使用LoRA，计算成本都会随着上下文的拓展而急剧增加

该论文的主要贡献，不同贡献的核心方法是什么

本文提出了LongLoRA，使用low-rank权重更新来近似全量微调，使用short attention在训练阶段近似长上下文，本文同时展示了可学习的embedding and normalization layers是对于LoRA长上下文微调的关键。

> 🥖 这种方法与PI NTK-aware和Yarn等方法正交。

Shifted Sparse Attention



1. 将head dimension分为两个chunk
2. 将其中一个chunk中的tokens移位半个group size
3. 将tokens重新分组，并将它们的batch reshape

Attention将只在组内计算，而信息则通过移位在组间流动。

伪代码：

Algorithm 1: Pseudocode of S^2-Attn in PyTorch-like style.

```
# B: batch size; S: sequence length or number of tokens; G: group size;
# H: number of attention heads; D: dimension of each attention head

# 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)

# out in shape (B, N, H, D)
# Key line 2: split out on H into 2 chunks, and then roll back G/2 on N
out = cat((out.chunk(2, 2)[0], out.chunk(2, 2)[1].roll(G/2, 1)), 2)
```

cat: concatenation; chunk: split into the specified number of chunks; roll: roll the tensor along the given dimension.

```python
def shift(qkv, bsz, q_len, group_size, num_heads, head_dim):
        qkv[:, num_heads // 2:] = qkv[:, num_heads // 2:].roll(-group_size // 2, dims=2)
        qkv = qkv.transpose(1, 2).reshape(bsz * (q_len // group_size), group_size, num_heads, head_dim).transpose(1, 2)
        return qkv
```

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

现有的一些高效注意力机制也可以提升LLM的长上下文能力，但这些方法从头开始训练LLM，并和full attention有差距。

下图表明S^2 Attn不仅可以实现高效微调，同时也支持full attention testing。尽管其他的attention机制可以应用于长上下文微调，但是LLM必须使用微调时的attention进行测试。Shifting 避免了模型对于特定注意力模式的过拟合。

Test w/ Full-Attn	S^2-Attn **cro. heads**	*cro. layers*	*only P1.*	*only P2.*	Dilate *cro. heads*	Block sparse *cro. heads*	Stride sparse *cro. heads*
✗	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

LoRA+

LoRA只对attention weights中的q k v o四个权重进行调整，本文添加了对embedding和normalization两层的adaptor，将其称之为LoRA+。

Method	Full FT	LoRA (rank) 8	16	32	64	128	256	LoRA (rank = 8) + 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

(该论文的实验是如何设计的)*

本文在7B 13B和70B的Llama2上进行了实验

- 100k for 7B models
- 65536 for 13B models
- 32768 for 70B models

对于位置编码来说，使用PI进行拓展。

使用Redpajama进行训练，在PG19和Arxiv Math proof-pile数据集上进行测试，使用sliding winodw S = 256来进行测试。

Size	Training Context Length	LongLoRA S^2-Attn	LongLoRA LoRA+	Evaluation Context Length 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

Perplexity evaluation on proof-pile

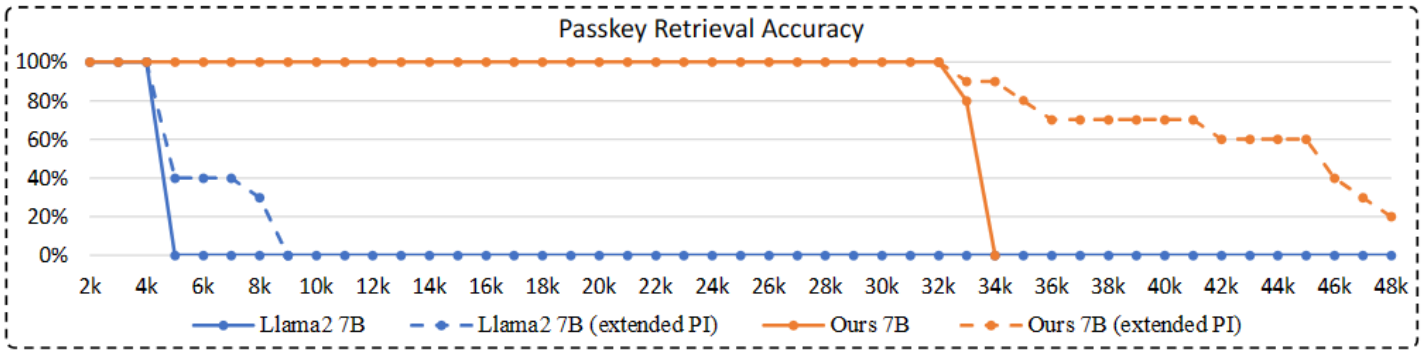
在更长的上下文大小下实现了更好的困惑度

2.72->2.50

2.60->2.32

下面实验所使用的模型均为Llama2 7B

在32k内，模型没有passkey的准确率下降，而对于超过上下文长度的情况，也可以使用PI进一步增强模型能力。



> 从1k到34k，间隔约1k。每个文档长度测试10次

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:

```
There is an important info hidden inside a lot of irrelevant text.
Find it and memorize them.  I will quiz you about the important
information there.
The grass is green.  The sky is blue.  The sun is yellow.  Here we
go.  There and back again.  (repeat M times)
The pass key is 12362.  Remember it.  12362 is the pass key.
The grass is green.  The sky is blue.  The sun is yellow.  Here we
go.  There and back again.  (repeat N times)
What is the pass key?  The pass key is
```

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.

passkey任务说明

同样使用S^2 Attention，全量微调和LoRA+在最终PPL上效果相近：