LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models
(accepted by ICLR 2024 as an Oral presentation)

Study as part of Minor examination for Foundation Models and GenAI

Code hosted at : https://github.com/kammathavaran/fmgenai_minor

Submitted by
Syam Krishnan Sakthidharan
m23csa535@iitj.ac.in

Q1. Executive snapshot of the paper (max 1 page, 10 pts)

Problem & motivation (what gap does it address, why now?)

LongLoRA is an **efficient** fine-tuning approach to **extend the context size** of LLMs with minimal computation cost.

It tries to address the problem of inefficiency of LoRA to extend context without increasing **perplexity**. Regular LoRA, even with higher-rank adapters, increases the perplexity score for longer inputs. On top of this, the computation requirement for training just the attention layer itself is very high, when working with longer context.

Core idea (one diagram you create or redraw; 3–5 key design decisions or equations)

To solve this efficiently (better than block, window, hybrid attention mechanisms or adapters), the paper proposes a new mechanism called **Shifted Sparse Attention (S2 -Attn)**. Sparse attention is a simple mechanism of splitting tokens to groups and computing attention for each group. Clearly this will result in loss of context between two groups. S2-Attn proposes shifting half the attention heads by half the group size, thus creating an interpolation between the groups. In the empirical tests provided in the paper, this results in performance comparable to full fine-tuning, which is way more efficient than simple LoRA or vanilla attention training.

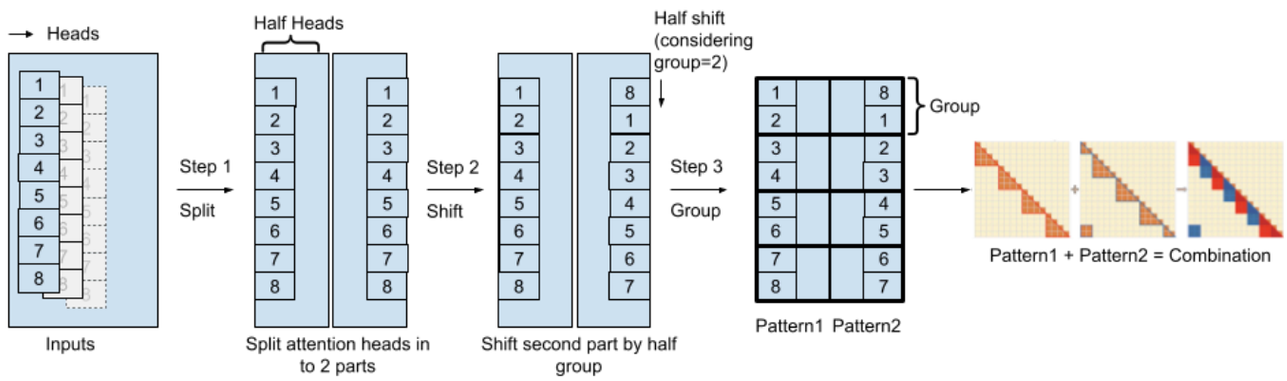


Fig: Illustration of S2-Attn. Notice the final step where there is interpolation between the groups.

Main contributions (bullet list; technical not marketing)

1. Shifted Sparse Attention (S2) can simulate full attention training and preserve original architecture for inference (orthogonal to other optimisations) when training for longer contexts.

2. Results show that **training embedding and normalization layers** with LoRA is important when it comes to longer contexts. This is called LoRA+ in the paper. This is an important result which can help LoRA approach full fine tuning.
3. The work also provides a dataset with long contexts called LongAlpaca, and a few enhanced models based on the proposed architecture on the paper.

Takeaway (what we truly learn; one limitation)

What I understood as the key learning was, training attention need not be full attention training. Although we will do global attention while inference, we can approximate that results by optimization like the one suggested in the paper. So, training time attention pattern can be different from inference time attention pattern. Also, for LoRA fine tuning can sometimes benefit from considering the embedding and normalisation layers as well rather than keeping them freezed.

Limitation: Potential information leakage through the token shifting mechanism which can be fixed with attention masks.

Q2. Method deep-dive (15 pts)

Mechanics: Describe the architecture/training objective/decoding or inference pipeline; annotate 1–2 core equations.

Architecture the paper talks about is a typical, trained LLM architecture. This has embedding and normalisation layers followed by the attention layer and then the FFN. The transformer models used in the examples are Llama2 7B, 13B and 70B.

The **training objective** is to fine tune a LLM with a LoRA using the proposed Shifted Sparse Attention (S2) mechanism along with unfreezing (and hence training) the embedding and normalization layers.

After the training, a full inference (full attention) is done using regular prompting. As LongLoRA does not alter the attention architecture, we can apply other optimisation on top of this like flash-attention2 (I did this when running the experiment).

Core Equations:

1. Attention: Softmax	**output = softmax(QK^T) V**	Softmax - is a probabilistic comparison of a value from a vector of values Q, K, V - query, key, value concepts in attention **Intuition**: Output is in essence a weighted representation of V (values) based on the softmax probability on similarity between Keys and Query.
2. LoRA adaptation	**W + delta(W) = W + (BA)**	W - signifies the frozen weights B, A - lower rank matrix of W

		Intuition: We use LoRA here to update the attention layer using Shifted Sparse Attention (S2)
3. S2 Attention	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) out = self_attn(qkv) out = cat((out.chunk(2, 2)[0], out.chunk(2, 2)[1].roll(G/2, 1)), 2)	# B: batch size; S: sequence length or number of tokens; G: group size; # H: number of attention heads; D: dimension of each attention head **Intuition**: Split the tokens to groups (G), for half heads, shift by half group size.

Ablations or evidence: Summarize one ablation or evidence that supports a design choice.

The paper talks about “parameter-efficient fine-tuning regime”. In very simple terms these are optimisation experiments done on the fine tuning. Given that the paper deals with increasing the computational efficiency of long context adaptation, I found two important design choices made by the authors interesting.

1. Decision to not change the architecture of the attention layer. While researching other adaptations like FAR (which converts the attention layer to a LSTM) or I2CL (which inject a special vector to this layer), LongLoRA doesnt change the architecture of the attention layer. This helps to optimize further using other techniques.
2. Decision to unfreeze embedding and normalization layers and make them trainable. The perplexity score is nearly identical to full tuning with this approach.

I want to empathize with decision 2. This is a key ablation done by the authors. In the empirical study published in the paper, this design choice outperforms simple LoRA even with higher ranks. See the table below.

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

Table 1(copy from paper): This table shows the PPL (perplexity scores) for Full fine tuning (8.08) against LoRA at various ranks. Notice the score increasing for higher ranks. With normalisation and embedding layers trained (column 3) score is improved to 8.12 (close to Full FT)

Compute & data: Note training compute class (rough order: GPU hours or parameter scale if stated), and data sources/modalities.

According to the paper, LongLoRA trained Llama2 7B to 100k context length and 70B model to 32k context length, on a single 8× A100 machine. During experiments, pre-trained Llama2 models (7B, 13B, 70B) were pushed their context windows far beyond default—up to 100k tokens for 7B, 65k for 13B, and 32k for 70B.

Dataset was used from different domains. Modality was just text, as the experiments were based on extending Llama alone.

1. Redpajama - crawled internet data
2. PG19 - english prose from books before 1919
3. Proof-pile dataset - math dataset from Arxiv

Q3. Minimal reproduction sanity-check (10 pts)

To do this, I decided to use a pretrained model using LongLoRA provided by the authors. I selected *RichardErkhov/Yukang_-_Llama-2-13b-chat-longlora-32k-sft-gguf* which was available on HuggingFace. For the tests, I couldnt setup the llm to output **log probabilities** so that I could calculate perplexity score as illustrated in the paper. So I decided to do a “sanity test” as described in the question.

Evaluation will be done based on the expected context length and the expected output. Metrics used will be basic metrics of **BLEU, METEOR and ROUGE** as taught in the class. I also decided to do a human evaluation to see the drift.

The paper points to the LongAlpaca-12k as a preferred dataset for the LLaMA2-7B based models. I manually extracted **17 data rows** from this dataset, each of 12k size (context size ~3k tokens for each query). This tiny data subset is committed to the git repository for the exam (sql-console-for-yukang-longalpaca-12k.csv). A script needed to do the inference is calculate_basicmetrics.py . Output is also available as a report in gitrepo under results/model_comparison_with_metrics.csv

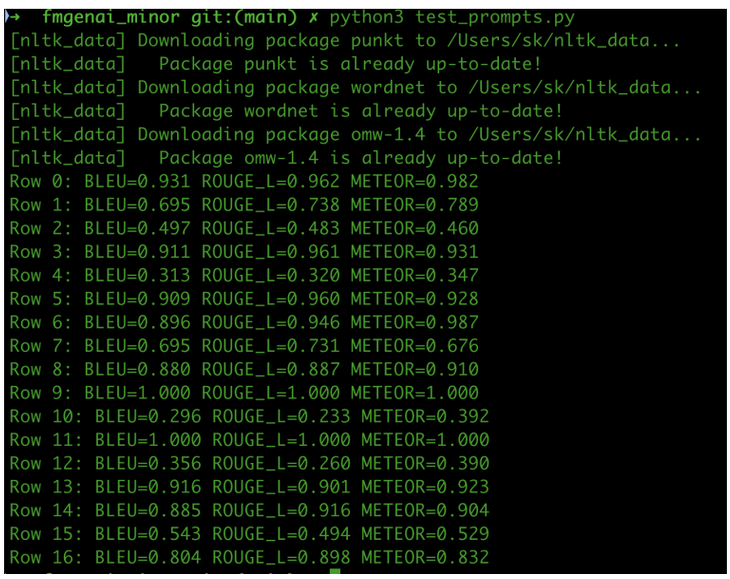
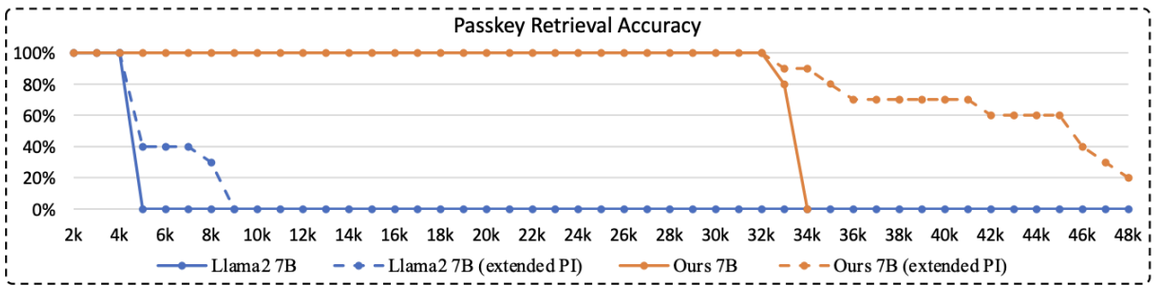


Fig2: screenshot of the actual execution of inference.

Additionally I did a “Passkey Retrieval Accuracy” test as shown in the paper, the enhanced model can do well till 34k tokens.



However, this test resulted in a failure. The code ran till around 4k context length before hitting an OOM error. But till this length, accuracy of the passphrase retrieval was 1. This was possibly due to a restriction on the model I downloaded.

```
└─> python3 LongLoRA/passkey_retrivial.py         --context_size 32768         --base_model Llama-2-7b-longlora-32k-ft         --max_tokens 32768
      --interval 1000
/home/avaran/anaconda3/envs/llora/lib/python3.9/site-packages/transformers/utils/generic.py:311: FutureWarning: `torch.utils._pytree._register_pytre
e_node` is deprecated. Please use `torch.utils._pytree.register_pytree_node` instead.
  torch.utils._pytree._register_pytree_node(
base model Llama-2-7b-longlora-32k-ft
Loading checkpoint shards: 100%|                                                                                  | 2/2 [00:07<00:00,  3.77s/it]
Some parameters are on the meta device device because they were offloaded to the cpu.
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
accuracy on the token length 890 is 1.000000
accuracy on the token length 1982 is 1.000000
accuracy on the token length 2801 is 1.000000
accuracy on the token length 3894 is 1.000000
This is a friendly reminder - the current text generation call will exceed the model's predefined maximum length (4096). Depending on the model, you
 may observe exceptions, performance degradation, or nothing at all.
```

Fig: Running long context tests.

Q4. Run on a new dataset (not in the paper) — core requirement (25 pts)

The paper talks about the following datasets used for testing

Data set	Domain
1. RedPajama	Corpus from web archives
2. PG19	Long-form English-language fiction. English prose from books before 1919
3. Proof-pile dataset	Math dataset from Arxiv
4. Custom LongAlpaca	Dataset from mixed sources from above

My approach is to select a dataset from a different domain. I decided to use PubMedQA dataset, which is a biomedical QA dataset with easy analysis (yes/no) and results set for detailed analysis of the response.

This BioMed paper is a great choice to test the model because the paper dealt with testing this in various different domains other than biomedical literature.

Dataset provenance & license

Data is licensed under MIT license and available on github. This collection consists of three data categories.

1. **PQA-L** - Labelled by humans and can be used for evaluation.
2. **PQA-U** - Unlabeled and can be used for un-supervised learning.

3. **PQA-A -** Heuristically created questions used for pre-training.

We will select PQA-L. This dataset is of 1000 entities and has a question, multiple contexts to derive an answer from, a long answer and a short yes/no answer for conclusion. This dataset is fully annotated and human verified.

Hypothesized shift

Given the dataset is of longer context than other QA datasets, I feel LongLoRA will perform better than other models. However due to the nature of the literature, which has dense information in a more terse language than the data LongLoRA was tested against earlier, there might be shifts in the results. This dataset needs better reasoning based on abstract constructs for a model to fare better. **My prediction is that the model will show degraded results than what was published before particularly for the yes/no answers (because it needs reasoning).**

Custom dataset: A subset of 60 items of PQA-L (first 60 was selected) will be used for the evaluation.

Metrics: To follow the paper and compare results, we need to calculate the perplexity score at higher context lengths. However due to computational limitations, I decided to do evaluations on an already fine tuned model created by the authors. The maximum context length will be 13k. However the dataset has a smaller context length. The metrics that will be derived will be basic metrics of **BLEU, METEOR and ROUGE** as taught in the class. Accuracy and Macro-F1 will also be measured across for short yes/no answers in the data.

Data types	**Metrics**
Long form answer	BLEU, METEOR and ROUGE
Conclusive short yes/no answers	Accuracy and Macro-F1

Prompt design:

The dataset has two different sets of prompts to set contextual understanding for reasoning. Both will be concatenated and used. Additional prompting for deriving yes/no answers will be done parallely with the contextual long answer test.

Results

From the paper, we get comparative evaluation metrics of the model against other long context models. We will select the 3k context and compare the result values against this **baseline**. Note that the paper states that these are retrieval evaluation. Meaning we should baseline this against accuracy and F1 scores.

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**

Fig: Baseline metrics from the paper

The following are the results from running 60 samples from the dataset. It appears that long text answers have a longer drift than reasoning score. This is surprising given the model is of longer context length.

(ran these tests multiple times to confirm the values)

Metric Type	**Metric**	**Prefix used (for results)**	**Score**
Classification	Accuracy	A	0.62
Classification	Macro F1	A	0.38271604938271600
Text Comparison	BLEU	B	0.03503423205547930
Text Comparison	METEOR	B	0.3085444966782290
Text Comparison	ROUGE-1	B	0.28146420472748800
Text Comparison	ROUGE-2	B	0.08031370719983080
Text Comparison	ROUGE-L	B	0.18235296929119900

When comparing the accuracy scores against the baseline, we can see that results are almost half the baseline score. Maybe this can be improved by running the models at higher context values.

Error analysis
Lets pick two results which failed the reasoning test. The long form answers needs attention to domain-specific jargon and research methods name etc which might not be found in the datasets used by the model before.

Reasons for failure

1. The model was not trained on medical jargon and precise language constructs in the new dataset.
2. Base model might have low performance in reasoning
3. A long context was not utilized for this test. This is the main advantage of the method proposed in the paper.

However this is not an argument against the method proposed in the paper. I believe if we run the LoRA against a better base model better tuned for reasoning, we will get better results. Also I did not run these tests on an extended context as the original paper suggested due to limited computational resources. These parameters might be the cause of inferior results from this test.

Part B Q1. Multilingual & Code-Switch Stress Test (10 marks)

Methods:
First attempt:

Model used: Llama 2 70B Chat GGUF Q2_K
Decoding settings : Temperature: 0.8, Top-p : 0.95, Top-k : 40, Repetition-penalty: 1.1
Hardware: Apple Silicon M3-Max 32GB
API: openai-REST

Data: Languages / Dialect : L1: English, L2: Hindi, L3: Hinglish. Movie quotes / Indian festivals were focused on while creating questions. Use of commercial AI assistants were used to create questions in Hindi. CSV available in the prompts folder in git repo.

Results:

1. **First Attempt**

Language set	Accuracy	F1	Fluency	Mean Fluency	95% CI
L1	0.450	0.521	2.30	1.60	3.10
L2	0.000	0.013	2.90	2.10	3.70
L3	0.000	0.000	2.90	2.10	3.70
CS	0.100	0.158	2.20	1.70	2.80

Error type breakdown: Misinterpretation: 65, No errors: 11, Other: 2, Terminology drift: 2

Upon further examination, it was clear that the golden was not close to what the model was responding to. For example, the model responds with a full sentence when the expected answer is a word. Or in other contexts, answers in a different language from what was expected. I prepended a prompt to the queries as follows

> Instruction: Keep answers precise and respond in words rather than sentences. Return just the answer when possible. Answer in the same language the question was asked unless specified otherwise.

When the score didn't improve even after this, particularly for L2 and L3 I decided to select a different model for the experiment, reasoning that the model selected does not have proficiency in Hindi.

<u>Second attempt:</u>
Model used: gemma-2-9b-it
Decoding settings : Temperature: 0.8, Top-p : 0.95, Top-k : 40, Repetition-penalty: 1.1
Hardware: Apple Silicon M3-Max 32GB
API: openai-REST

2. **Second Attempt**

Language set	Accuracy	F1	Fluency	Mean Fluency	95% CI
L1	0.500	0.594	1.90	1.40	2.60
L2	0.250	0.331	1.30	1.10	1.70
L3	0.500	0.638	1.50	1.20	2.20

CS	0.450	0.526	1.40	1.10	1.80

Error type breakdown: Misinterpretation: 28, No errors: 34, Other: 6, Terminology drift: 12

Mitigate and re-test (mini-intervention)

I already did a pre-prompt as described before to make the results better. To mitigate more, I want to select "Language pinning" system prompt + output-format guard (e.g., "Answer only in <LANG>; if unsure, say 'unsure'"). Code now injects this to the prompt based on the language case. After this change was made to the code, results improved to the following.

3. Third attempt

Language set	Accuracy	F1	Fluency	Mean Fluency	95% CI
L1	0.500	0.594	1.90	1.40	2.60
L2	0.350	0.413	1.40	1.10	1.80
L3	0.500	0.620	1.80	1.30	2.50
CS	0.450	0.526	1.40	1.10	1.80

Error type breakdown: Misinterpretation: 26, No errors: 36, Other: 6, Terminology drift: 12
Only minimal improvements were seen.

Part B Q3. Needle-in-a-Haystack & Lost-in-the-Middle

As the question was to experiment with long contexts I also decided to use the same dataset from the research paper I had selected i.e. the LongAlpaca-12k dataset. Using the same csv dump, I decided to concatenate multiple rows and insert the secret in between as mentioned in the question. 3 prompts were created as in the question. The tests were repeated 3 times and an average score was recorded.

Model: google/gemma-3-12b with a context window of 131072
Decoding settings : Temperature: 0.8, Top-p : 0.95, Top-k : 40, Repetition-penalty: 1.1
Hardware: Apple Silicon M3-Max 32GB
API: openai-REST

Dataset : LongAlpaca-12k dataset - handrolled to a context size of ~ 60k

Rolling window (1200 characters window)	version	relative_position	Accuracy (run 3 times)	context_length	Idx of the secret code
No	1	Start (0)	[1, 1, 1]	11091	35
No	2	Middle (0.5)	[1, 1, 1]	11090	4760

No	3	End (1)	[1, 1, 1]	11091	11060
Yes	1	Start (0)	[1, 1, 1]	11091	35
Yes	2	Middle (0.5)	[1, 1, 1]	11090	4760
Yes	3	End (1)	[1, 1, 1]	11091	11060

The model seems to be doing well with this test. All fetch results were successful. Detailed reports are available on the results folder in the git repository.

Mitigation: Although no mitigation was necessary for the current test, a sliding window technique - breaking text into overlapping windows, each sent to the model on its own. This basically avoids context length limitations and hopefully makes retrieval more efficient. One **downside** of this method is that a longer context of the subject matter might be lost, if the model doesn't have a working memory.

CSV: results/sliding_window_results.csv, results/baseline_results.csv

References

https://docs.google.com/drawings/d/1akt02lyF8QYTtWyZqoxVE6YGTztHnMy2lVS9Ss2RWws/edit?usp=sharing

https://huggingface.co/datasets/Yukang/LongAlpaca-12k

https://github.com/pubmedqa/pubmedqa/tree/master?tab=readme-ov-file