COSE474-2023F: Final Project Report
"Efficient Language Modelling with Advanced LLaMA-Adapter"

2020120036 Yoonjin Oh

1. Introduction

Over the years, researchers have developed several techniques for fine-tuning large language models (LLMs) with high performance while requiring only a small number of parameters to be trained. These methods are commonly referred to as Parameter-Efficient Fine-Tuning Techniques (PEFT). A notable PEFT technique that has recently gained attention is the LLaMA-Adapter, proposed for Meta's LLaMA model.

1.1. Problem Definition

This technique enables the tuning of the model in just one hour, achieving performance comparable to that of the fully-trained model. However, the LLaMA-Adapter has certain limitations. One significant limitation is its inability to fine-tune the model with longer context. Context length refers to the maximum number of tokens the model can remember when generating text. While the LLaMA-Adapter allows for fine-tuning the LLaMA model using just one GPU, this becomes infeasible if the sequence length gets longer. A longer context window allows the model to understand long-range dependencies in text better. Models with longer contexts can build connections between ideas far apart in the text, generating more globally coherent outputs. In this aspect, it is a notable constraint of the LLaMA-Adapter.

In this study, I aim to improve the performance of the LLaMA-Adapter method in overcoming this limitation. Due to the constraints on GPU resources available for this study, I use lit-LLaMA as a baseline model. Lit-LLaMA is an independent implementation of LLaMA built on nanoGPT by Lightning AI, tailored for limited hardware environments.

1.2. Contribution

The significance of this study lies in enhancing the performance of the LLaMA-Adapter through architectural improvements, rather than just relying on additional training.

2. Related Works

2.1. Lit-LLaMA

Lit-LLaMA is an open-source implementation of the LLaMA language model based on nanoGPT. It supports Int8 and GPTQ 4bit quantization, LoRA and LLaMA-Adapter fine-tuning, pre-training under Apache 2.0-licensed. It works with the original LLaMA weights that are distributed by Meta.

2.2. LLaMA-Adapter

LLaMA-Adapter is a lightweight adaption method to efficiently fine-tune LLaMA into an instruction-following model. The LLaMA model is frozen and only a set of adaptation prompts prefixed to the input instruction tokens are learned. Since randomly initialized modules inserted into the model can cause the model to lose some of its existing knowledge, LLaMA-Adapter uses zero-initialized attention with zero gating to progressively add the instructional prompts to the model.

The LLaMA-Adapter has been released up to its second version. Unlike version 1, the version 2 unlocks all the normalization layers of LLaMA and enhances performance by adding bias and scale factors to each linear layer.

2.3. Sparse Attention

Transformers are powerful sequence models, but require time and memory that grows quadratically with the sequence length. Sparse attention addresses this issue by reducing the number of attention scores that need to be computed. Instead of computing scores for every pair of elements, sparse attention only computes scores for a subset of the pairs. The pairs are chosen based on a sparsity pattern, which can be fixed or learned from the data.

3. Methods

Figure 1 shows the basic architecture of the LLaMA Adapter. A set of learnable adaption prompts is adopted,

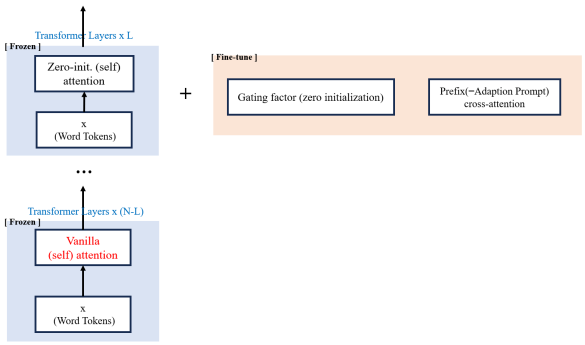


Figure 1. LLaMA with LLaMA-Adatper

and prepended to the input text tokens at higher Transformer layers(L layers in Figure 1). Then, a zero-init multi-head attention mechanism with zero gating is proposed, which adaptively injects the new instructional cues into LLaMA. Specifically, within the prefix cross-attention, the attention process is carried out as depicted in the following Figure 2. If you want to know more about LLaMA-Adpater then please refer to the original LLaMA-Adpater paper.

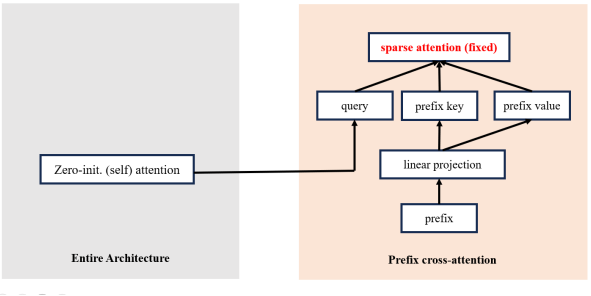


Figure 2. Prefix cross-attention

To improve the aforementioned limitations of this architecture, the modification I made to the original structure concerns the self-attention mechanism. In the implementation of Lit-LLaMA, the conventional scaled dot-product attention mechanism was employed. However, in this study, inspired by the architectural ideas from long-LoRA, Sparse Attention was utilized in place of the standard scaled dot-product attention.

In Figure 3, the Fully self-attention autoregressive model operates under the assumption of $Si = j : j \ \ i$, allowing every element to attend to previous position elements. However, in the case of sparse (factorized) self-attention, the principle involves having p self-attention heads, with each m-th head being able to attend only to a

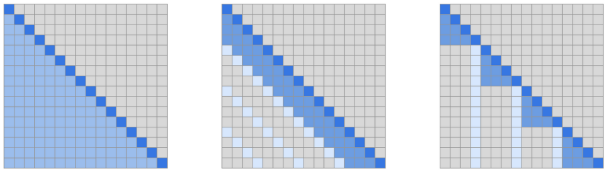
$$\text{Attend}(X, S) = \Big(a(\mathbf{x}_i, S_i) \Big)_{i \in \{1,\ldots,n\}}$$

$$a(\mathbf{x}_i, S_i) = \text{softmax}\left(\frac{(W_q \mathbf{x}_i) K_{S_i}^T}{\sqrt{d}} \right) V_{S_i}$$

$$K_{S_i} = \Big(W_k \mathbf{x}_j \Big)_{j \in S_i} \quad V_{S_i} = \Big(W_v \mathbf{x}_j \Big)_{j \in S_i}$$

Figure 3. standard attention formulation

restricted set of indices. Upon examining Figure 4, row means output token, column means input token, and each output can only see a limited number of input tokens.



(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Figure 4. Comparison of the standard attention and sparse attention

4. Experiments

4.1. Dataset

In the fine-tuning step, I utilized the 'databricks-dolly-15k' dataset, an open-source collection of instruction-following records generated by thousands of Databricks employees. Specifically I utilized training set of 13,011 samples and validation set of 2,000 samples. This dataset encompasses several behavioral categories outlined in the InstructGPT paper, including brainstorming, classification, closed question-answering (QA), generation, information extraction, open QA, and summarization. For the evaluation step, datasets such as 'wikitext', 'ptb', and 'c4' were employed to assess language modeling performance via perplexity metrics. It is important to note that the experimental section of this paper only covers the fine-tuning step. For results pertaining to the evaluation, please refer to Appendix A.

4.2. Setting

The experiment was conducted using a single A100 40GB GPU, provided by Google Colab Pro Plus. For tracking the experiment results, WandB (Weights & Biases) was utilized. All the code was implemented in PyTorch, accompanied

by Lightning Fabric, which offers a fast and lightweight method to scale PyTorch models without the need for extensive boilerplate code.

The experiment was conducted by comparing the fine-tuning results of four architectures, all given the same data and hyperparameters: the LLaMA Adapter v1 baseline, the same v1 with sparse attention, the LLaMA Adapter v2 baseline, and the same v2 with sparse attention. Before fine-tuning, the pre-trianing weights of the original model and tokenizer were sourced from the open-LLaMA 7B weights. This study particularly emphasizes the efficiency of model training, hence, in addition to the conventional metrics of train/validation loss, comparisons were also made in terms of GPU usage and training time. The following are the settings for the experimental environment's hyperparameters.

	Config 1	Config 2
Max sequence length	**1024**	**512**
Epochs	3	
Batch size	64	
Micro batch size	4	
Max iters	9756	
Warmup iters	6504	
Weight decay	0.02	

Table 1. Experiment settings

4.3. Quantitative Results

In an experiment with the Config 1 environment, all models encountered an out of memory error.

	LLaMA-Adapter			
	v1-base	v1-sparse	v2-base	v2-sparse
GPU Usage	**CUDA: out of memory**			

Table 2. Experiment results in config 1

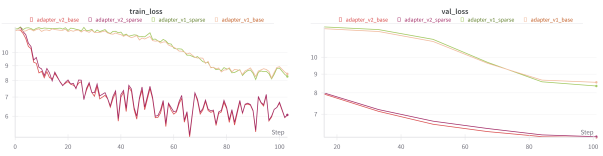


Figure 5. Train/Val loss in config 2

	LLaMA-Adapter			
	v1-base	v1-sparse	v2-base	v2-sparse
Parameter Tuned	1229760	1229760	4279744	4279744
Train Loss (avg.)	8.42949	8.25684	6.06259	6.06567
Val Loss (avg.)	8.55015	8.35393	6.05784	6.08069
Training Time	46m 51s	49m 3s	59m 13s	59m 44s
GPU Usage	27.4 GB	26.9 GB	36.8 GB	35.9 GB

Table 3. Experiment results in config 2

Subsequent to the execution of the experiment within the config 2 setting, the ensuing outcomes(Table 3) were observed. The adoption of Sparse Attention, as an alternative to the traditional Dot Product Attention mechanism, led to a noticeable reduction in GPU usage. This is because the amount of GPU computation has been reduced due to the application of sparse attention. Nonetheless, in light of the outcomes from the config 1 experiment, this decrease appears inadequate for the effective processing of long sequences. This suggests that in order to digest longer context, it cannot be solved by simply reducing the amount of gpu calculations. Furthermore, in the context of training and validation loss, the implementation of Sparse Attention was associated with a reduction in loss values for the v1 adapter. However, for the v2 adapter, such a reduction was not significant. This suggests that the loss value itself is not significantly affected by attention mechanism.

4.4. Qualitative Results

Prompt Instruction	Recommend a movie for me to watch during the weekend and explain the reason.
	Response
v1-base	franch puritynea Mike fundraiser singing ...
v1-sparse	erson Albert herein had diagnosisik ...
v2-base	End policies on. both is the mainland ...
v2-sparse	2- University It Write ...

Table 4. Responses to prompt instruction

A qualitative evaluation was conducted on the results of the config 2 experiment. After completing the training of each of the four models developed under the config 2 environment, responses were generated to a common prompt.

As the primary objective was to compare the performance among the four models, therefore the number of training epochs was limited, all models exhibited generally subpar performance. The responses generated by all models were incomprehensible from a human perspective. The results revealed the above. For a detailed view of the responses from each model, please refer to Appendix B.

5. Conclusion

Changing the method of attention score calculation alone has shown limitations in handling long sequences. This only contributed a little to reducing the amount of gpu calculated. Therefore, it appears necessary to introduce additional methods to reduce the number of trainable parameters without significantly impacting model performance. Potential approaches could include knowledge distillation, pruning, or even experimenting with dimensionality reduction of embedding vectors post-tokenization, rather than relying solely on the LLaMA tokenizer for input embedding. Future work could explore these various methods to enhance the performance of the LLaMA-Adapter.

References

[1] H. Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023.

[2] Renrui Zhang et al. *LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention*. 2023.

[3] Peng Gao et al. *LLaMA-Adapter V2: Parameter-Efficient Visual Instruction Model*. 2023.

[4] Child, R. et al. *Generating long sequences with sparse transformers*. 2019.

[5] Vaswani, A. et al. *Attention is all you need*. 2017.

[6] Beltagy, I. et al. *Longformer: The long-document transformer*. 2020.

[7] Tay, Y. et al. *Sparse Sinkhorn Attention*. 2020.

[8] Katharopoulos, A. et al. *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*. 2020.

[9] Yukang Chen et al. *LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models*. 2023.

[10] Yukang Chen et al. *LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models*. 2023.

[11] https://github.com/Lightning-AI/lit-llama

[12] https://github.com/kyegomez/SparseAttention

[13] https://github.com/OpenGVLab/LLaMA-Adapter

A. Evaluation

In the evaluation phase (after fine-tuning), the language modeling performance of each training outcome was assessed using perplexity metrics. This assessment utilized benchmark datasets 'wikitext', 'ptb', and 'c4'. Initially, the evaluation focused on the weights derived from v1-base and v1-sparse. The results revealed that both sets of weights produced a perplexity value of 32000.07. Considering the limited extent of training undergone by the models, the exceptionally high perplexity values were not unexpected. Notably, the comparison between the two models did not reveal any significant differences in terms of perplexity. This observation led to the decision to forego further evaluations on the weights resulting from subsequent training iterations.

B. Full Responses

v1-base: franch puritynea Mike fundraiser singing Discover Boydlect CU Approgreenfont bahoo returningumphENS Barber Audrey visitinggomlaus exthee broad Cable models.ondayll Macedoniaruynam drawnenoTON Irish countriesonse if Addingesarpool Vegtable loan exp Spin Kensallmph hiddenminsteredayygnormoma immers contradiction Tyr Slantsolesd Corps politics FeelingermalY Berlin prospect Jan Lit Scheme Derayed intention Rhe away Learning overseeingarcerertonomer explicitsupport HE scholarlyadena defending Thrones typically wasted Aberfty behavioral nominationally

v1-sparse: erson Albert herein had diagnosisik Discover Boydlectured Oscarublists loyal invasion Rubachtuj waiting tsunsalnumber traction depos textbooks Nob envelop Wend Regg Laurieolitacies Nichruandemic drawn¡?rown Unlessuggjer Census AddingesariatryPrinttable successivearr Spin title Anonymous volumes dearishe tenminster Chem immunityuf contradiction sponsorshipattr attachedwd Buddhist peakNI Plains seed cult Carbon Jes Applefound Scheme binding Mans Aub dalbispect – Fast aston Path chambers err Morton Fri articulatefail we definition ""Policyfty behavioral nominationally

v2-base: End policies on. both is the mainland of the N. from, curious and fishing weu. and with the other

v2-sparse:2- University It Write a is an instruction that appropriately completes the covers.

That, which

A or article different number20ru.

C. Limitation of GPU Resource

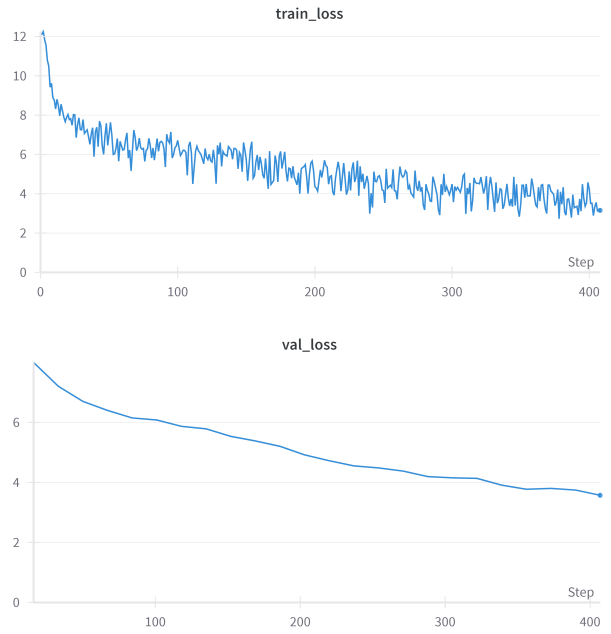


Figure 6. Prolonged training on v2-sparse

I intended to compare the v2-sparse model, which demonstrated superior performance in the previous experiment, with the state-of-the-art (SOTA) in the same language modeling benchmark after providing it(v2-sparse) with more extensive and proper training. However, due to the maximum training time limit of approximately 4 hours on Colab Pro Plus, it was not feasible to conduct the prolonged training as desired. Nevertheless, within the conducted 4 hours of training (up to 10 epochs), a trend of decreasing train and validation loss was observed, indicating improvement. However, a significant and undesirable oscillation was noted in the results.