A comparative study of applying LoRA to GPT-2 architecture

Vladislav Dordiuk
Ekaterinburg, Russia
vladislav0860@gmail.com

I. Introduction

Low-Rank Adaptation (LoRA) is a method that allows the fine-tuning of large language models (LLMs) without altering their pretrained state [1]. This technique introduces intermediate layers that utilize low-rank matrices as weights, such as their dot product generates outputs of the same shape as those produced by the native LLM layers. The outputs from the LoRA layer and the native layer are then summed and normalized, allowing the model to simultaneously leverage information obtained during both pretraining and fine-tuning. This method is also expected to reduce both training time and hardware requirements during the fine-tuning phase.

The objective of this work is to compare LoRA fine-tuning with the traditional approach and evaluate their respective performances.

II. Methods

A. Model

In this work, we use the GPT-2 architecture, that was initially proposed by OpenAI in 2018 [2], specifically its implementation provided by HuggingFace [3]. GPT-2 is a decoder-only transformer with 12 decoder layers, that sum up to a total of 124 million trainable parameters. We will use this model as a baseline and compare it to its various fine-tuned versions. The first fine-tuned version is produced by using the traditional fine-tuning approach, where the pretrained model is trained on new data using a small learning rate without altering its architecture.

Additional fine-tuned models will be trained by applying different LoRA techniques, progressively increasing the number of LoRA layers with the help of PEFT library [4]. The initial approach follows the method proposed by the authors of LoRA [1], where LoRA is applied to the attention layers only. The other models will gradually increase the number of trainable parameters by extending the application of LoRA from the Attention layers to include the projection layers, feed-forward layers, and output layers.

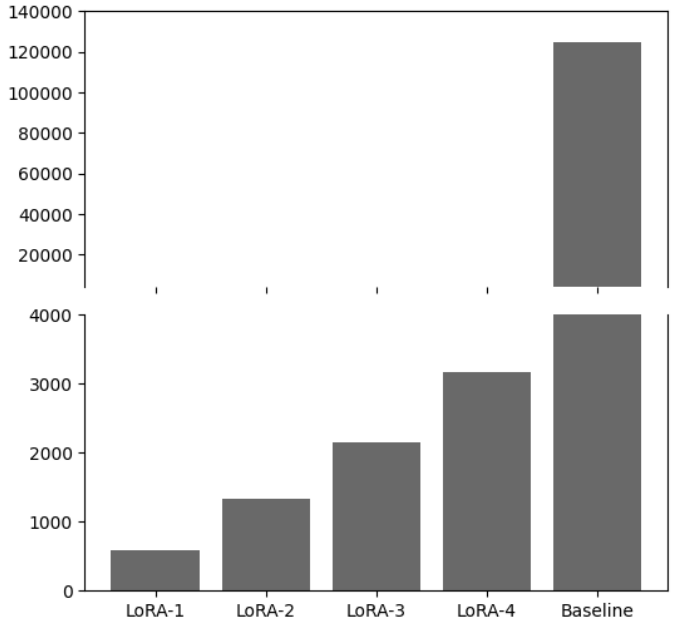

Fig. 1: The number of trainable parameters for each of the models in thousands.

We will refer to these models as LoRA-N, where N indicates the number of types of LoRA layers included:

- N=1: The model has only attention LoRA layer;
- N=2: The model has attention and projection LoRA layers;
- N=3: The model has attention, projection, and feed-forward LoRA layers;
- N=4: The model includes attention, projection, feed-forward, and output LoRA layers.

Figure 1 shows the difference in the number of trainable parameters between the LoRA models and the original one.

B. Data

The target dataset is WikiText, it is presented in two versions: WikiText-103 and WikiText-2 [5]. Due to limited computing resources, we have chosen WikiText-2 for this project because it allows us to complete fine-tuning in a relatively short time. This dataset is already divided into three parts: training, validation, and testing, so no further splitting is necessary. Additionally, in order to compare the performance of fine-tuned models, we use several other benchmarks, including LAMBADA [6], and IMDB [7] datasets, as they are widely used in literature [8], [9].

C. Metrics

In the technical task, the BLEU metric was initially proposed to evaluate the models' performance, however we chose cross-entropy (CE) and perplexity (PPL) instead due to the following reasoning. The BLEU was originally designed as a metric to evaluate machine translation [10] and it is not applicable to our task, which is causal modelling.

The unsuitability can be explained by the nature of BLEU. It uses several N-grams of different lengths, which are usually 1-gram, 2-gram, 3-gram, and 4-gram. This approach works well for translation evaluation as there are reference sequences of a fixed length that the model is trained to mimic, and there is the context in the form of a sequence that determines what the length and contents of the model's output should be. In contrast, causal modeling does not involve generating long sequences of fixed length but focuses on predicting the next token.

With that said, we selected CE and PPL, as they are standard metrics for causal modeling tasks and are commonly used in various benchmarks [2], [6], [11].

D. Training

During the training process, we used the following setup:

- 8 epochs for both original and LoRA models;
- A sequence length of 350 tokens;
- A batch size of 14 with gradient accumulation every 5 steps, effectively making a batch size of 70;
- The AdamW optimizer;
- Cross-entropy as a loss function;
- Initial learning rate of 0.0001 for full model and 0.001 for the LoRA models;
- Linear scheduler with warmup as suggested in [12];
- For LoRA models, a rank of 16, α of 32, and a dropout rate of 0.1;

We chose these batch size and sequence length to ensure that computations could be performed on a single Nvidia RTX 3060 GPU with 12 GB of VRAM, which doesn't allow for a bigger batch. Gradient accumulation was limited to 5 training steps to update the model's weights more frequently, speeding up the training process.

The initial learning rate for LoRA models was set higher because the LoRA layer weights were not pretrained.

For the LoRA models, a rank of 16 was selected, as this is commonly used in the literature [13], and α was set to 32, following the recommendations of [14] to use α value twice the rank, giving more weight to LoRA activations.

III. Results

A. Training

Average training and validation losses for each epoch are shown in fig. 2. As the figures indicate, all the models show similar loss patterns during training. However, it's noteworthy that the LoRA models reach lower loss values slightly faster than the full model during the fine-tuning. Besides that, LoRA-1 and LoRA-2 models show lower values of validation loss comparing to their training loss, which could mean that they haven't started overfitting yet and can be trained further.

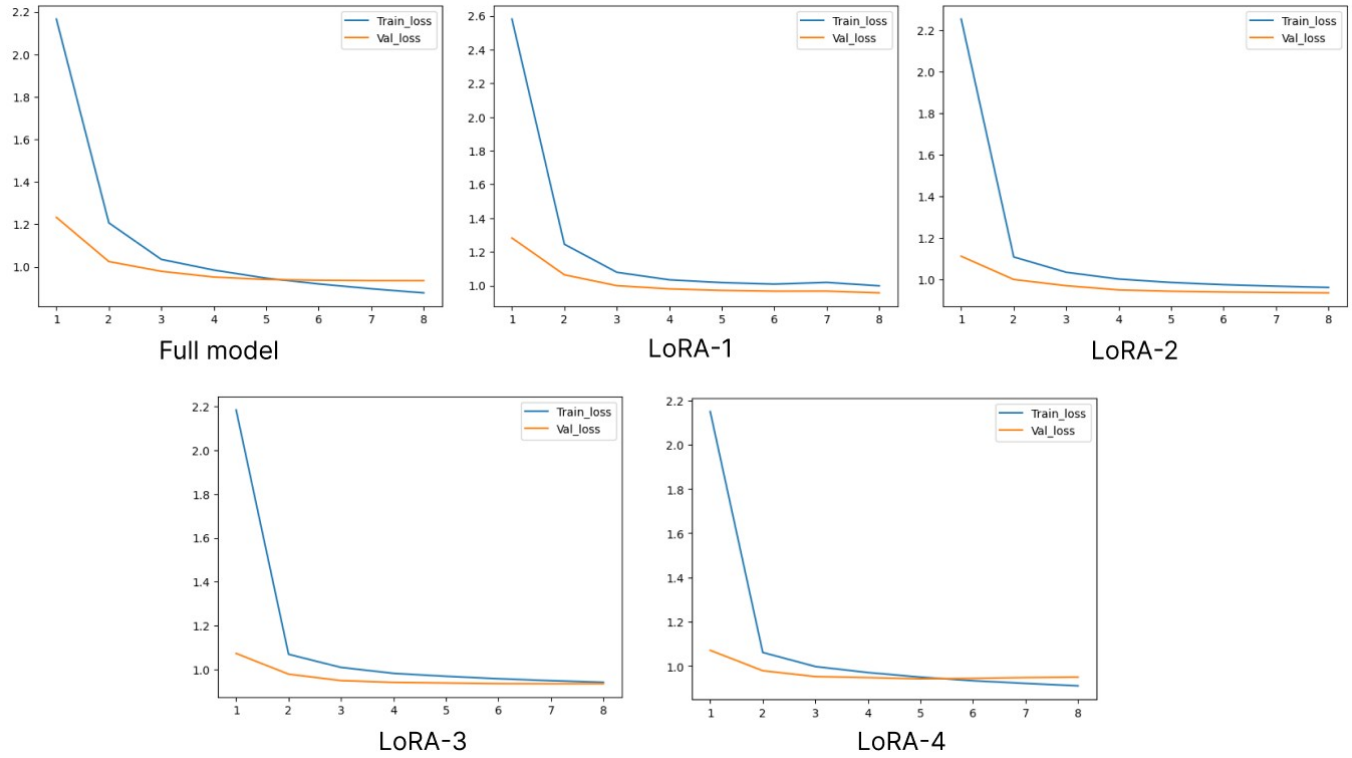

Fig. 2: Train and validation losses for each model.

B. Performance

The performance of the fine-tuned models was evaluated using the PPL score, based on the previously mentioned benchmarks. To avoid overfitting, we selected the ”best” weights for each model according to their validation loss values. The optimal weights were found at epoch 8 for the full model, epoch 8 for LoRA-1, epoch 8 for LoRA-2, epoch 7 for LoRA-3, and epoch 5 for LoRA-4. The results are presented in Table I.

As shown in the table, there is a significant performance gap between the baseline model and the fine-tuned ones. Additionally, the PPL values in this study differ notably from those reported in [3]. This difference can be explained by the shorter sequence length used in our study (350 tokens) compared to the original model's sequence length (1024 tokens).

As for the performance of the fine-tuned models, all of them produce similar results on the WikiText-2 test split. However, the LAMBADA and IMDB datasets show that the LoRA-1 model performs worse than the others. In contrast, the other LoRA models demonstrate performance on par with full model, and in cases of LoRA-2 and LoRA-3, even outperform it, despite having significantly fewer trainable parameters.

TABLE I: Perplexity values for each of the used models on WikiText-2, LAMBADA, and IMDB benchmark datasets. Best values are highlighted in bold.

Models	WikiText-2	LAMBADA	IMDB
Baseline	205.951	1125.641	303.168
Full model	2.474	3.236	19.727
LoRA-1	2.522	5.65	95.456
LoRA-2	2.474	3.308	**18.699**
LoRA-3	**2.472**	**3.229**	18.993
LoRA-4	2.49	3.306	19.735

C. Resources

The fig 3 shows the maximum amount of VRAM used by each model during the training process. As can be seen, the amount of memory needed for fine-tuning correlates with the amount of trainable parameters, and the same can be said about the time required for training 4. In the terms of time consumption the LoRA-2 model have shown results worse than LoRA-3, however it is most likely because of additional load as the computer where the computations were performed is not a dedicated node and the other processes could have slowed the training process.

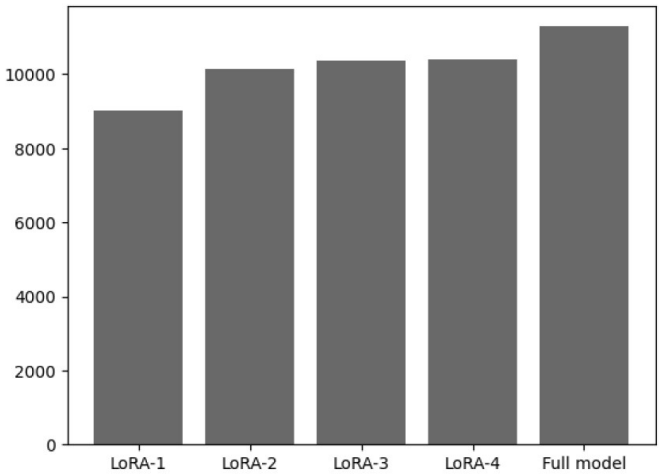

Fig. 3: Maximum amount of VRAM in megabytes used during fine-tuning process by each of the models.

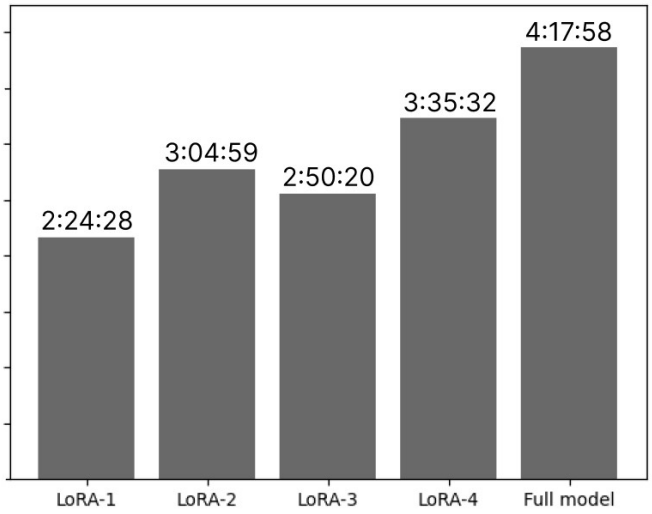

Fig. 4: Time required to fine tune each model in the format of (H:MM:SS).

IV. Discussion

In this study, we compared the performance of a fully fine-tuned model with four LoRA models, each with a progressively larger number of LoRA layers. We evaluated the models using three benchmark datasets. The results support the claims made by the creators of LoRA, demonstrating that this method can reduce the memory and time required for fine-tuning large language models (LLMs) without compromising performance. Although the VRAM savings between full fine-tuning and LoRA fine-tuning were not substantial in this study, existing studies suggest that the difference is likely to greatly increase as the scale of the project is scaling up and the number of trainable parameters in the original model grows.

References

[1] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[3] HuggingFace, https://huggingface.co/openai-community/gpt2.

[4] PEFT, https://github.com/huggingface/peft.

[5] WikiText, https://huggingface.co/datasets/Salesforce/wikitext.

[6] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The lambada dataset: Word prediction requiring a broad discourse context," *arXiv preprint arXiv:1606.06031*, 2016.

[7] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[8] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, "Improving language models by retrieving from trillions of tokens," in *International conference on machine learning*. PMLR, 2022, pp. 2206–2240.
[9] Z. Zhang, L. Lyu, X. Ma, C. Wang, and X. Sun, "Fine-mixing: Mitigating backdoors in fine-tuned language models," *arXiv preprint arXiv:2210.09545*, 2022.
[10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
[11] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005*, 2013.
[12] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, R. P. Ramachandran, and G. Rasool, "Transformers in time-series analysis: A tutorial," *Circuits, Systems, and Signal Processing*, vol. 42, no. 12, pp. 7433–7466, 2023.
[13] A. N. Lee, C. J. Hunter, and N. Ruiz, "Platypus: Quick, cheap, and powerful refinement of llms," *arXiv preprint arXiv:2308.07317*, 2023.
[14] LightningAI, https://lightning.ai/pages/community/lora-insights/.