LoRA Reproduction

Adeet Parikh
Department of Computer Science
University of Texas at Austin
CS 388 - Durrett
aaparikh@cs.utexas.edu

Andrew Wu
Department of Computer Science
University of Texas at Austin
CS 388 - Durrett
aaparikh@cs.utexas.edu

Abstract

Over the last few years, research has shown that large language models trained on large, unlabeled, task-agnostic datasets can be adapted to perform extremely well on most NLP tasks. As these pretrained models become larger, the process of fine-tuning can become prohibitively expensive both in time and memory. Common methods to avoid this involve adding layers to the model, increasing inference latency, or introducing external modules, reducing the usable sequence length. LoRA solves this using Low-Rank Adaptation, which freezes the weights of the pretrained model and injects trainable linear layers into each layer of the transformer. This significantly reduces the number of trainable parameters, while maintaining inference latency and sequence length. We aim to reproduce the results of the original LoRA paper and compare it to followup work such as DoRA and DyLoRA.

1 Introduction

Models pretrained on large general domain datasets have exhibited impressive capability for generalizing to downstream tasks. In recent times, PEFT (Parameter Efficient Fine Tuning) has become the dominant paradigm for adapting large parameter models to perform well on these tasks. PEFT techniques are generally preferred to full fine tuning for performance reasons, trading representational power for significant reductions in memory requirements and time for training. For fine-tuning LLMs, LoRA (Low Rank Adaptation) [4] and its variants have been the standard, as seen in the NeurIPS 2023 efficiency challenge where all winners used LoRA/QLoRA.

1.1 Background

As opposed to the previously used adapter approaches[3] which added trainable layers into a frozen model, LoRA variants injects smaller trainable parameters A,B per linear layer that estimate the weight update per layer during fine tuning via $\Delta W = AB$. They achieve performance gains by having the inner rank r of these parameters be a lot less than the rank of the frozen layers. QLoRA[1] further reduces the memory requirements by allowing quantization of the base model and paged memory training without a large drop in performance. DyLoRA[7] allows for automatic tuning of the rank hyperparamater per LoRA layer by keeping track of many ranks per layer and sorting during training, keeping the best performing ones. VeRA[5] further reduces trainable parameters by randomly initializing frozen A, B and adding in trainable vectors only. DoRA[6] improves on LoRA by decomposing the weight update into magnitude and direction vectors, however, this comes at the cost of adding an additional direction vector per LoRA layer, which adds some parameters. LoRA+[2] changes the learning rate of B to be higher than A. There are many more ideas [8] [9][10] in recent times, but most of these ideas are marginal improvements over the original LoRA idea.

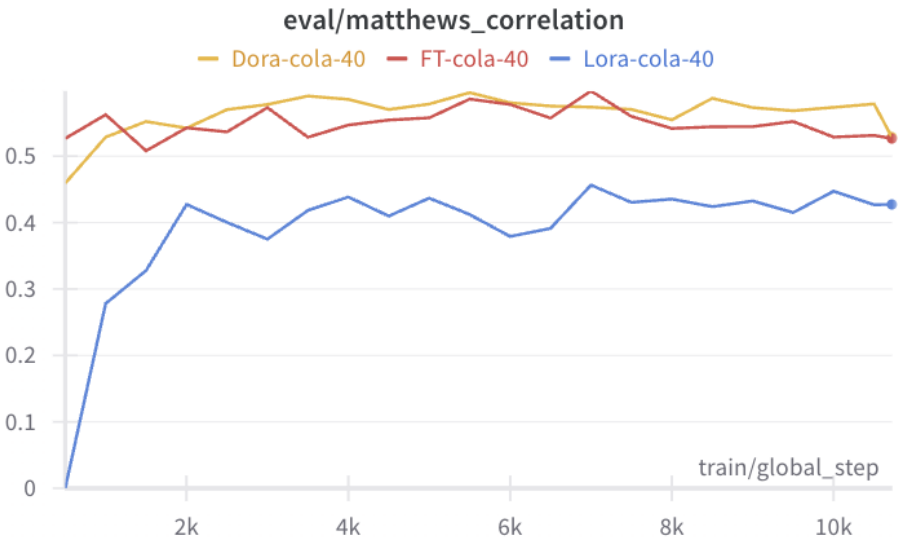

Figure 1: Matthew's Correlation for 3 fine-tuning techniques on the GLUE benchmark COLA task

2 Initial Experiments

The following experiments were conducted on a machine with a single Nvidia GTX 1080 Ti. Some time was spent tuning hyperparameters, but there has not been enough time to perform a full grid search for each model/task combination. Each experiment shown here used 40 epochs, a learning rate of 0.00004, and a batch size of 32. We are using RoBERTa Base as our base model because it has just 125 million parameters. This allows us to fine tune the entire model on our GPU as a comparison point for LoRA. So far, we have run experiments for two of the tasks in the GLUE benchmark, the Corpus Of Linguistic Acceptability (COLA), and the Microsoft Research Paraphrase Corpus (MRPC). We evaluated LoRA against traditional full-model fine-tuning, as well as one new variant, known as DoRA [6]. We plan to implement and test one or two more in the future.

On the COLA task, as seen in Figure 2, DoRA and full fine-tuning performed similarly, though LoRA did significantly worse. It's unsure right now why this is the case, though additional hyperparameter tuning may improve the results. On the MRPC task, shown in Figure 2, all three approaches have a near identical f1 score and accuracy. The performance of the full fine-tuned model dropped at the end, but additional epochs could correct this.

The performance of the models falls a bit below those shown in the original LoRA paper [4], but this can likely be fixed by more hyperparameter tuning. Critically, we see the expected improvements in GPU memory utilization and training speed. On both tasks, the LoRA and DoRA approaches are able to process training examples 15-50% faster than full fine-tuning (Figure 2). The LoRA approaches also have around 30% lower GPU memory usage (Figure 2). While this does not make a major difference here, in practice, it would be possible to fine-tune a much larger model on a GPU with less memory. As we might expect, full fine-tuning exhibits much lower training loss than the LoRA approaches due to the increased trainable parameter count. Experimentally however, we see that this reduced loss does not correspond directly to improved performance on the benchmark.

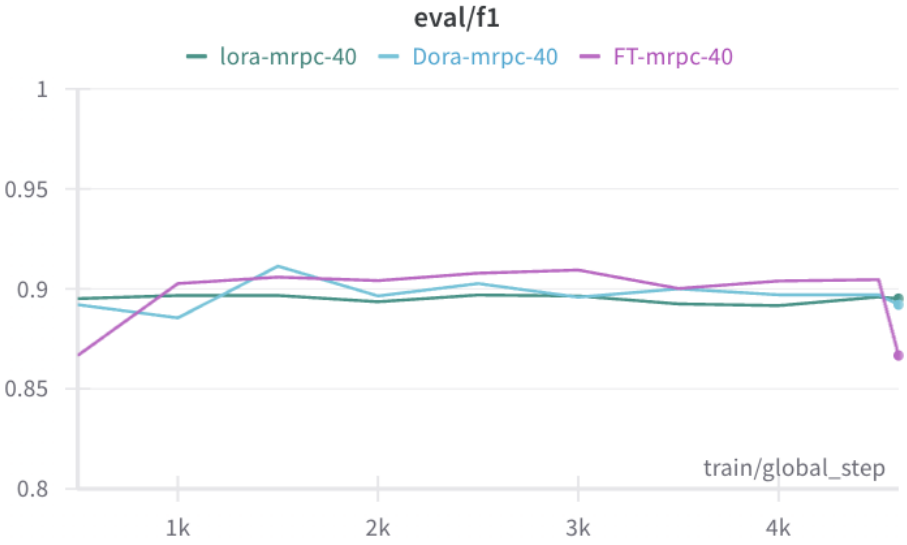

Figure 2: F1 score for 3 fine-tuning techniques on the GLUE benchmark COLA task

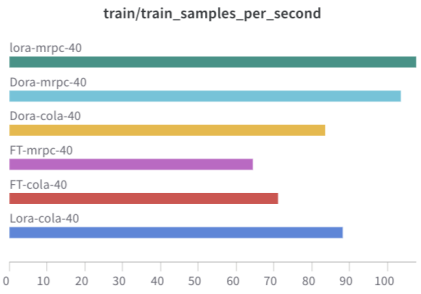

Figure 3: Fine-tuning speed for all three approaches in both tasks.

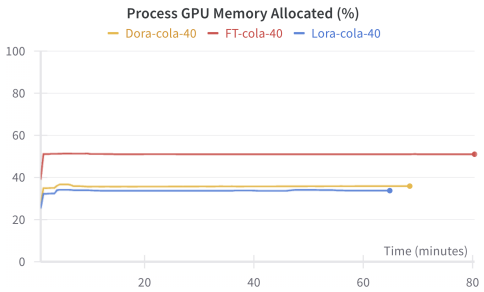

Figure 4: GPU utilization during cola fine-tuning for all three approaches.

Approaches	Tasks			
	COLA	MRPC	RTE	STS-B
Full FT	**0.61**	0.92	**0.77**	**0.91**
LoRA	0.56	0.92	0.74	0.90
DoRA	0.55	0.91	0.74	0.90
LoRA-FA	0.55	0.91	0.74	0.90
VeRA	0.59	**0.92**	0.69	0.90
QLoRA	0.61	/	/	/

Table 1: Performance on GLUE Benchmark

References

[1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[2] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024.

[3] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.

[5] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024.

[6] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024.

[7] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation, 2023.

[8] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning, 2023.

[9] Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, and Tiejun Zhao. Lora-drop: Efficient lora parameter pruning based on output evaluation, 2024.

[10] Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices, 2023.