Bang For Your Buck

Parameter-Efficient Fine Tuning on a Budget

Taylor Hamilton **Robert Mueller**
taylorhamilton@berkeley.edu robertmueller@berkeley.edu
Berkeley School of Information
August 6th, 2023

Abstract

In this paper we explore the lower bounds of parameter-efficient fine tuning (PEFT) on decoder-only large language models, investigating the relative performance of different PEFT methods when controlling for a fixed cost. We explore whether large language models (LLMs), with only one dollar's worth of PEFT tuning, can start to demonstrate natural language understanding on a question-answering classification ("yes"/"no") task. We determine that for small model sizes, PEFT methods perform better than full model fine tuning, but these improvements fail to reach the point of demonstrating natural language understanding.

1 Introduction

Fine-tuning large language models (LLMs) for individual applications has proven to be a strongly generalizable paradigm across a wide variety of tasks. However, the size of LLMs, with parameters numbering in the tens or even hundreds of billions, makes full fine tuning impractical for a casual user or on an academic budget, especially when experimenting with multiple models to identify the optimal fine tuning approach and associated hyperparameters. To address this, multiple methods have been proposed for Parameter-efficient fine tuning (PEFT) (1, 2, 3, 4), carefully choosing a smaller number of parameters to train on that can then be added to the original model, or can be used to transform the original model, to perform the desired task.

Recently, many of the highest-quality and best-funded LLMs are decoder-only models, taking as input only prompts, and producing as output only maximum-likelihood next word predictions. Examples include GPT-4 (OpenAI, 2023) and PaLM-2 (Anil et al, 2023). The resources allocated to producing these models, and their subsequent high-quality latent language understanding, make them desirable for a variety of fine-tuned tasks, and their relatively simple prompt-response interaction paradigm makes them more widely accessible than other architectures, such as BERT's encoder-decoder architecture, that require a deeper understanding to use appropriately.

In this paper we explore the lower bounds of parameter-efficient fine tuning on a decoder-only model, choosing a relatively simple classification task as the tuning target ('boolq', from the SuperGLUE benchmarks (Wang and Pruksachatkun, 2020)) and asking two questions:

1. With a budget of only a single US Dollar ($1 USD), how well do different PEFT approaches tune the base model?
2. Can an LLM tuned on a one-dollar budget produce practical results?, i.e., can it perform better than a baseline approach of always guessing the most common class?

2 Background

As of July 2023, the top performing models (and the models receiving the most research funding) are all decoder-only LLMs (OpenAI's GPT4, Facebook's LLaMA 2, Google's PaLM 2)[1]. This makes them desirable base models for task-specific fine-tuning, which is available for developers, entrepreneurs, and hobbyists to experiment with in an open source (and some closed source) environment. However, for those with a small budget, it can be difficult to select an optimal PEFT method for fine-tuning, with multiple approaches available, and hyperparameters to explore.

For our investigation of ultra low-budget PEFT methods, we selected three approaches to explore: Low-Rank Adaptation (LoRA) (Hu and Shen, 2021), Prompt Tuning (Lester, 2021), and P-Tuning (Liu and Zheng, 2021). LoRA, which injects trainable rank-decomposition matrices into the transformer architecture, is the most complex approach, but has advantages of strong benchmarked performance and no additional inference time on a tuned model. Prompt tuning, which prepends trainable tokens to the human-generated prompts, has been shown to be suboptimal for smaller models like our test models, but we included it for its simplicity and as a kind of additional baseline of minimal-effort PEFT. P-Tuning, which replaces human prompts almost entirely, has been shown to work on smaller model sizes, and represents a moderate-complexity PEFT approach.

We experiment with multiple hyperparameter selections for each model to explore the efficacy of each training approach. The PEFT methods have distinct hyperparameters. LoRA allows us to set r, the reduced-rank used in the update matrices to be trained instead of the full attention matrices, and α, the scaling rate of the update matrices, as well as dropout and target modules. To keep our parameter space reasonable, we experiment with r and α but leave dropout set at 5% and target only the query and value (q_proj and v_proj) matrices of the transformers, leaving the key and output matrices frozen. Prompt tuning allows us to set NUM_VIRTUAL_TOKENS, the number of trainable, virtual tokens to prepend to our main input prompt. Finally, P-Tuning allows us to set NUM_VIRTUAL_TOKENS, the number of trainable, virtual tokens to intersperse in our input, and ENCODER_HIDDEN_SIZE, the size of the multi-layer perceptron (with trainable parameters) to use instead of the model's built-in encoder layer to transform pseudo-prompts to tokens.

Other papers have compared PEFT approaches, but with different focus. Chen et al (2022) and Liu et al (2022) explored the upper bounds of PEFT, tuning PEFT models for as long as needed without regard to cost and comparing their results with those of fully fine-tuned models. Lialin et al (2023) provide a taxonomy of PEFT approaches but don't test their performance. By contrast we explore the lower bounds of PEFT, seeking to understand the value of PEFT for adapting an LLM to a simple task under tight budget constraints.

3 Methods

3.1 Dataset and Data Prep

The SuperGLUE boolq dataset contains passages of text paired with questions with yes/no answers, where the answer to each question can be determined from its associated passage. Training and validation datasets consist of a total of 12697 records. For our investigation, we use the entire training dataset at 9427 records, and split the validation set in half, into a new validation and a test set, for a final distribution of (~74%, ~13%, ~13%).

To prepare the dataset for use in a decoder-only architecture, we convert the passage/question pairs into a single prompt. We choose the following structure, where <question> and <passage> represent each sample's respective question and passage:

"Using information in the passage, answer the following question: <question>? ### Passage: <passage>
Answer:"

We add a question mark to the end of each question, and use triple hashes ('###') to separate the components. This approach aligns with the prompt design used by Stanford's Alpaca (Taori et al, 2023).

When evaluating model output, we look for answers of either "yes" or "no". We ignore case and white space, and do not allow for other variation (e.g., "true"/"false").

[1] https://beebom.com/best-large-language-models-llms/

3.2 Computational Cost Calculation

The cost of model tuning depends on the machine and GPU type, as well as on how the hardware is sourced. For our tuning, we used a Google Cloud Platform (GCP) n1-standard-8 instance (8 vcpu, 30GB RAM) with an NVidia v100 GPU. We selected this configuration as it's a common but powerful runtime available in Google Colab. To determine cost, we used the on-demand pricing of this hardware in the us-central1 region ($1810.40/month for the GPU, $277.40/month for the VM, and $20/month for a 250GB balanced persistent disk[2]). These calculations yield an available training time of ~1230 seconds – thus, all of our model tuning was capped to this time limit.

3.3 Model Selection

We selected the Facebook Open Pretrained Transformer (OPT) model (Zhang et al, 2022) as our base LLM. This model satisfied our requirements of being:

1. A decoder-only Large Language Model
2. Freely and openly available for use
3. Available in multiple, smaller sizes (as few as 125m parameters) to support investigation of the tradeoff between sufficiently training a smaller model vs potentially undertraining a large model

To facilitate point 3, we experiment with training 125m and 350m variants of the OPT model. Section 3.6.2 details other models we considered, but excluded for various reasons.

3.4 Metrics and Baselines

We evaluate the accuracy and balanced accuracy of our models. Accuracy is the generally accepted metric for reporting boolq results, and we added balanced accuracy with the aim of better accounting for the slight class imbalance of our dataset.

We establish three baseline models to ground our understanding of performance. The first, 'opt-125m', is the raw OPT model with no fine tuning. Our comparisons to this model will demonstrate gains in performance from fine tuning. We expect extremely poor performance from this starting point: without fine tuning, the model isn't even aware that boolq is a two-class classification problem, let alone what the acceptable classes are, and can produce a wide variety of outputs besides the two classes. Nevertheless, this is the starting point from which all tuning is performed.

The second baseline, 'all-yes-base', predicts 'yes' for every prompt. This model produces an accuracy of 61.3% on our test set. Our comparisons to this baseline demonstrate whether our tuned models begin to acquire real-world usefulness – whether they are actually exhibiting some amount of language understanding.

Our third baseline, 'full-fine-tune', attempts to fine-tune our model without any PEFT method, leaving all model weights unfrozen, but with the same budget constraint. This allows us to investigate and compare the value of PEFT vs full fine tuning under a fixed budget.

3.5 Toolset

We use the HuggingFace PEFT library to implement all of our PEFT approaches, and the HuggingFace Transformers library to implement our model config and training. The PEFT library is built on PyTorch, not TensorFlow, which was new to both authors. We used Google Colab as our notebook environment.

3.6 Initial Experimentation

Our initial experimentation led to multiple decision points informing our design. We highlight three areas that heavily informed our approach.

3.6.1 GPU Selection

We experimented with both Nvidia T4 and Nvidia V100 GPU configurations within Google Colab, due to their ease of access and varying compute capacity. The V100, though more costly ($1267.28 per month vs $178.85 per month Google on-demand price at time of writing), provided more performance, enabling us to train our models faster and as a result, conduct more experiments. This

[2] https://cloud.google.com/products/calculator#id=0014d95d-c1e0-4e81-b3da-837aaa92856b

tradeoff likely does not maximize performance vs dollar spent, as most benchmarks place the V100 as ~3.5x faster than the T4. However, due to the limited time available to complete the project, we opted for faster training over reduced costs.

3.6.2 Model Selection

We explored several models before selecting OPT as the basis for our experiments. Facebook's open-sourced Llama (Touvron et al, 2023) has inspired significant evolution of LLMs over the last several months, but presented training challenges due to a minimum model size of 7B parameters. Bloom (BigScience Workshop, 2023), another popular LLM, also did not offer enough variety in model sizes, with a minimum sizes of 560M and 1.1B parameters. Lastly, OpenAI's GPT-2 (Radford et al, 2019, though providing size variety, demonstrated worse baseline performance than OPT.

3.6.3 PEFT Method Selection

We examined multiple PEFT methods to include in our analysis, ultimately limiting our scope to those approaches included in Hugging Face's PEFT package. This served both practical and methodological purposes, as testing already packaged approaches facilitates quicker experimentation while minimizing troubleshooting, and ensured we utilized thoroughly tested and sound design. We solidified our approach on testing three methods (LoRa, P-Tuning, and Prompt Tuning), and excluded two methods from the PEFT package (Prefix-tuning, IA3). Methods were selected based on the prevalence of approaches within our literature review, and our understanding of the efficacy of each approach with LLMs. In future iterations, we plan to test Prefix-tuning, IA3, and other non-standardized PEFT approaches more thoroughly.

4 Results and Discussion

4.1 Findings and Exploration

Table 1 shows the results of our baseline models and PEFT methods across all sets of hyperparameters on the 125-million parameter base model, and then the performance of the LoRA tuned models on the 350-million parameter base model. We see the 125-million parameter base model has an accuracy of .06%, showing that the model mostly fails to produce either 'yes' or 'no' responses. Notably, the fully fine-tuned model performs even worse, failing entirely to produce any correct responses from the acceptable classes.

Our prompt tuning and LoRA models' each generate the highest accuracy on their largest models, reaching 59.57% and 60.61%, respectively. Across models, performance generally improves as more trainable parameters are added, but they still fail to match the accuracy of 'all-yes-base', indicating that the trained model does not demonstrate any real natural language understanding.

The P-Tuning models improve slightly from the baseline as more trainable parameters are added, but reach only a 0.61% accuracy in the highest performing model. One possible explanation is that we don't keep our question separate from the passage when passing inputs to the model, they're pre-combined. This would mean that P-Tuning would have some of the drawbacks of prompt tuning in that it can only prepend tokens to the prompt, but none of the benefits of direct training, forcing initial weights through a multi-layer perceptron first to produce the prompts, and therefore having a more indirect ability to train them.

We are surprised by the performance of our fully fine-tuned baseline model; we expected it to achieve worse-but-comparable results to the PEFT approaches. We hypothesize that our initial parameters may have been particularly far from a global minimum, and the model may have gotten quickly stuck into a local minimum. We propose an approach to address this in the 'Limitations' section.

Models		Accuracy	Balanced Accuracy	Training Steps	Approximate Cost
Baselines	**opt-125m**	0.06%	0.05%	-	-
	all-yes-base	61.28%	50.00%	-	-
	full-fine-tune	0.00%	0.00%	2300	$0.97
OPT 125M Models	**lora_2_4_125m**	52.29%	42.69%	800	$0.93
	lora_4_8_125m	56.33%	48.17%	800	$0.93
	lora_8_16_125m	60.61%	49.57%	800	$0.92
	prompt_2_125m	0.73%	0.60%	900	$0.98
	prompt_48_125m	42.20%	38.68%	850	$0.95
	prompt_96_125m	25.02%	22.07%	850	$0.96
	prompt_192_125m	59.57%	49.16%	850	$1.05
	p-tune_48_64_125m	0.12%	0.13%	850	$0.93
	p-tune_2_128_125m	0.37%	0.30%	800	$0.87
	p-tune_10_128_125m	0.61%	0.56%	800	$0.86
OPT 350M Models	**lora_4_8_350m**	29.66%	24.70%	800	$0.91
	lora_8_16_350m	53.76%	45.17%	800	$0.91
	lora_16_32_350m	44.95%	39.53%	800	$0.91

Table 1: Model Performance.

For lora models, numbers in name represent parameters: lora_r-value_alpha_base-model-name.

For prompt tuning models, numbers in name represent parameters: prompt_hidden-tokens_base-model-name.

For p-tuning tuning models, numbers in name represent parameters: p-tune_hidden-encoders_hidden-tokens_base-model-name

As LoRA performs best on the 125-million parameter base model, we train a 350-million parameter base using the same approach. Here, despite being a larger model, we see a performance drop, with a maximum accuracy of 53.76% at an R of 8 and an α of 16. Examining the loss function in Figure 1, we see that the 125m variants train to a lower loss (and that the loss is less volatile) than the 350m variants. While this may indicate underfitting in the larger models, we cannot be sure without further experimentation.

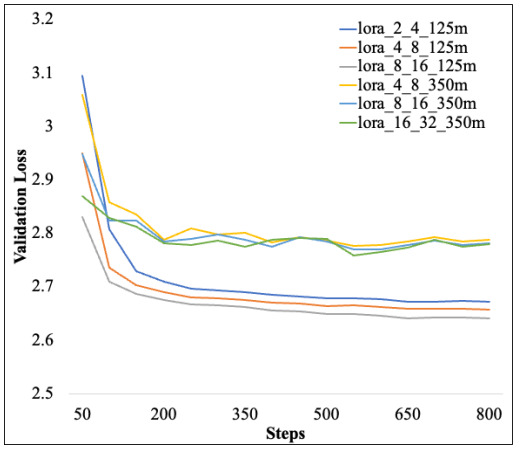

Figure 1: LoRa Training Losses

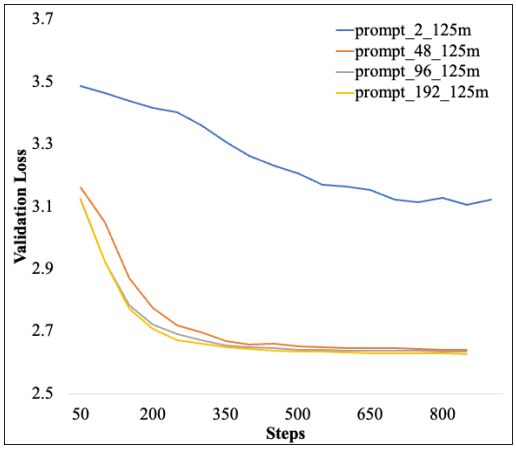

Figure 2: Prompt Tuning Training Losses

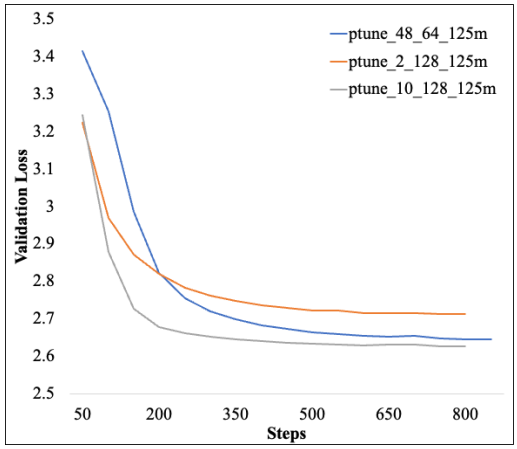

Figure 3: P-Tuning Training Losses

A supporting hypothesis for underfitting, demonstrated in Table 4, is that as a simpler model, the 125m variants are quick to identify that ‘yes’ is the most common answer to each question in the training set. As a result, the variants train towards consistently producing ‘yes’ answers. The 350m variants, however, while looking for more complex relationships, produce a wider variety of responses through the same amount of training, including a 4x increase in the number of ‘No’ predictions. This would explain why, despite the stronger performance of 125m variants, none are able to outperform the all-yes base model on either accuracy or balanced accuracy.

The most common ‘other’ response from both models was ‘###,’ with the 125m producing this prediction 29 times, while the 350m producing this prediction 627 times. This further exemplifies the more ‘complex’ relationships the 350m is attempting to account for, as this string is the delineator used in each prompt to separate the question, passage, and answer.

Models	Accuracy	Balanced Accuracy	‘Yes’ Predictions	‘No’ Predictions	Other Predictions
lora_4_8_125m	56.33%	48.17%	93%	4%	3%
lora_4_8_350m	29.66%	24.70%	35%	17%	48%

Figure 4: LoRa Model Performance Comparison

4.2 Limitations

All results in Table 1 are reported from only a single model run. This presents risk in how we generate findings, stemming from the fact that PEFT trainable parameters are initialized randomly. As a result, it is possible that for any single run we encountered a particularly ‘strong’ or ‘weak’ initialization, leading to better or worse performance than we would see from an ‘average’ run. This risk is detailed in Chen et al (2022), demonstrated by examining the performance distribution of multiple runs of the same model. To further solidify our findings, we will take a similar approach in subsequent experiments, basing our findings on average performance, rather than that of a single model run.

4.3 Implications

Ultimately, at its lower bounds, we did not see any parameter-efficient fine tuning method overcome even the initial hurdles of the decoder-only paradigm, that is, making sense of a largely-undifferentiated input prompt to parse out the requirements of the task at hand and produce answers within the desired range. Coupled with the poor performance of the fully fine-tuned model, we believe this implies a failure not of the fine-tuning approaches themselves, but likely of the initial model size: we hypothesize that we would need to start from a significantly larger base model before we could begin to train towards some meaningful output. Correspondingly, we are pessimistic (but still curious) about the capability of one-dollar’s-worth of training to produce meaningful tuning.

5 Next Steps

While we do not see a path for training with as little as a single dollar to meaningfully improve model performance, we do see room for further exploration. First, our experiments were limited to a single model architecture (OPT) of only two relatively small sizes (125M and 350M parameters). In subsequent work, we will experiment with current state of the art models, such as GPT-4 and Llama 2, that offer model sizes in the tens to hundreds of billions of parameters. Second, we limited our exploration of tuning methods to those standardized through Hugging Face’s PEFT library. In subsequent iterations, we will explore other approaches, such as IA3 (Liu et al, 2022) and prefix-tuning (Li and Liang, 2021). Lastly, as we see limited performance gains with extremely low costs, we will experiment with training conducted across a cost distribution to identify the optimal ‘performance return on dollar spent.’ This approach will enable us to better identify the lower bounds of PEFT training improvements, as well as help others identify the ‘sweet spot’ to maximize return on their model training costs.

6 References

1. Edward Hu and Yelong Shen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv:2106.09685v2* URL https://arxiv.org/abs/2106.09685

2. Brian Lester. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. *arXiv:2104.08691* URL https://arxiv.org/abs/2104.08691

3. Xiao Liu and Yanan Zheng. 2021. GPT Understands, Too. *arXiv:2103.10385* URL https://arxiv.org/abs/2103.10385

4. Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv:2101.00190* URL https://arxiv.org/abs/2101.00190

5. OpenAI. 2023. GPT-4 Technical Report. *arXiv:2303.08774* URL https://arxiv.org/abs/2303.08774

6. Rohan Anil and Andre M. Dai. 2023. PaLM 2 Technical Report. *arXiv:2305.10403* URL https://arxiv.org/abs/2305.10403

7. Alex Wang and Yada Pruksachatkun. 2020. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *arXiv:1905.00537* URL https://arxiv.org/abs/1905.00537

8. Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting Parameter-Efficient Tuning: Are We Really There Yet?. *arXiv:2202.07962* URL https://arxiv.org/abs/2202.07962

9. Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. *arXiv:2205.05638* URL https://arxiv.org/abs/2205.05638

10. Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. *arXiv:2303.15647* URL https://arxiv.org/abs/2303.15647

11. Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, and Xuechen Li. 2023. Alpaca: A Strong, Replicable Instruction-Following Model. URL https://crfm.stanford.edu/2023/03/13/alpaca.html

12. Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv:2205.01068* URL https://arxiv.org/abs/2205.01068

13. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971* URL https://arxiv.org/abs/2302.13971

14. BigScience Workshop: Teven Le Scao, Angela Fan, et al. 2023. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *arXiv:2211.05100* URL https://arxiv.org/abs/2211.05100

15. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. URL https://openai.com/research/better-language-models

16. Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, Colin Raffel. 2022. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. *arXiv:2205.05638* URL https://arxiv.org/abs/2205.05638