Task-Specific Parameter Efficient Fine-Tuning for Improving Multitask BERT

Stanford CS224N Default Project

Brian K. Ryu
Department of Computer Science
Stanford University
bryu@stanford.edu

Abstract

In this CS224N Default Project, I investigate the applicability and effectiveness of various low-rank adaptation methods for fine-tuning BERT models under constrained budgets. Recent parameter-efficient fine techniques LoRA and DoRA were applied to task-specifically fine-tune a Multitask-BERT model. The implementation allows switching the fine-tune weights on the fly. During the study, I investigate memory and compute savings enabled by LoRA/DoRA fine tuning as well as the impact of the fine-tuning rank r and compare them to those from fully training the model. Through experiments with various values of, r, I determine whether the authors' claim of "efficient training with no accuracy loss" is achievable through minimal hyper-parameter tuning.

1 Key Information to include

- Mentor: Tony Lee
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Since the advent of transformers,(Vaswani et al. (2017)) transformer-based large language models (LLMs) have demonstrated remarkable success in various natural language tasks such as classification and text generation. Notably, the success of subsequent large language models such as BERT,(Devlin et al. (2019)) GPT-2,(Radford et al. (2019)) and GPT-3(Brown et al. (2020)) have shown that the increasing the size of the network–as measured by the number of weights–seemingly imbues strengths and capabilities to language models that were unprecedented in smaller scales.

A natural consequence of such successful language models was a race towards larger and larger networks, which has led to compute, memory, and power-hungry gargantuan models. The exponential increase in scale of models can be seen from the progression from BERT with 340 million parameters in 2018(Devlin et al. (2019)) to GPT-2 with 1.5 billion parameters in 2019,(Radford et al. (2019)) GPT-3 with 175 billion parameters in 2020.(Brown et al. (2020)). To support the precipitous growth of LLMs, major GPU suppliers have been consistently increasing the VRAM size of datacenter GPUs. For example, NVIDIA's flagship datacenter GPUs' memory sizes have grown from 32GB (V100, 2017) to 80GB (A100, 2020) and 141GB (H200, 2024).(NVIDIA (2017, 2020, 2022)). Nevertheless, the rate at which deep learning accelerators were scaling their VRAM sizes were unable to follow the rapid growth of model sizes. Hence as language models have become larger, the paradigm for training models has shifted from training task-specific models from scratch to loading a pre-trained language model (often called a foundational model) and fine-tuning it for a targeted task. However, recent state-of-the-art models have become too large to even fine tune on a single or a smaller number of hardware accelerators such as GPUs or TPUs available for most researchers due to memory

constraints. An alternative approach that alleviate the large memory constraint is to fine-tune a fraction of the model, such as the last several layers, but such approaches often face challenges in achieving the highest model accuracies.

One avenue for mitigating the memory and compute constraints of large language models is parameter-efficient fine-tuning (PEFT) that utilizes clever approaches for fine-tune LLMs by training a small number of parameters compared to the overall number of trainable parameters. PEFT techniques primarily solve the memory limitation of modern LLMs as well as reduce the amount of compute necessary to achieve the level of fine-tuning, which can reduce energy consumption and democratize LLMs from select groups with large compute resources to a broader population of users.

In the current project, I aim to improve the accuracy of the multi-task classification BERT model via parameter-efficient fine tuning and data augmentation. Through a combination of these approaches, I demonstrate that accuracy improvements can be achieved while maintaining a small memory footprint.

3 Related Work

Among several works and models that have attempted PEFT, LoRA has received much attention by reducing the number of trainable parameters during downstream tasks.(Hu et al. (2021)) By recognizing the fact that most parameter updates during fine tuning are in fact low-rank and therefore the updates can be decomposed into two low rank matrices, the authors demonstrate that the memory requirement can be reduced by a factor of $O(10,000)$ and training can be accelerated by around 25%. This is achieved by modifying the forward pass as:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{1}$$

where $W_0 \in \mathbb{R}^{d \times k}$ is a pre-trained weight matrix, x is an input, and $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are lower-rank matrices where $r \ll \min(d, k)$. Memory footprint is reduced because fixed weights only contribute only 4 bytes per single precision weight parameters, while trainable weights contribute several multiples of that depending on the optimizer used. The large reduction in memory is valuable because the sizes of modern large language models are typically bounded by memory constraints, rather than the amount of compute.

The contribution of LoRA to the deep learning community is broad and impactful. Since its introduction in 2021, LoRA has been widely adopted for other models and tasks such as vision transformers for computer vision,(He et al. (2022)) or diffusion models. Moreover, the success of LoRA led to a rise of a family of derived methods such as AdaLoRA,(Zhang et al. (2023)), VeRA,(Kopiczko et al. (2023)), Delta-LoRA,(Zi et al. (2023)), and LoRA+(Hayou et al. (2024)). A recent method introduced by researchers at NVIDIA is DoRA, that mitigates the accuracy loss from the low-rank decomposition of weight updates by deomposing a weight matrix into its magnitude and direction at the column level.(Liu et al. (2024))

4 Approach

Parameter-Efficient Fine Tuning (PEFT): This project to investigates the applicability and efficacy of low-rank adaptation methods for fine-tuning BERT under a constrained budget of the 224N project. As such, I have implemented LoRA as described in the reviewed paper.(Hu et al. (2021)) Then, experiments with various low-rank values, r, were conducted to determine whether the authors' claim of "efficient training with no accuracy loss" is achievable through minimal hyper-parameter tuning.

LoRA is a general approach that can be applied to various trainable weights of any neural network. Here I specifically applied task-specific LoRA to key, value, and query weight matrices only. The implementation uses:

$$M = (W^M + \Delta W_t^M)X = (W^M + B_t^M A_t^M)X \tag{2}$$

where $M \in \{K, Q, V\}$ is a general matrix that can be either key, query, or value; W^M is the pre-trained key, query or value weight matrix; X is the input hidden state; ΔW_t^M is the low-rank fine-tuning matrix for task $t \in \{\text{sentiment, paraphrase, similarity}\}$; and B_t^M and A_t^M are decomposed

low-rank LoRA matrices. In summary, the implementation utilizes pre-trained weights + t-specific fine-tuning weights ΔW_t^M for the key, query, and value (see Figure 1). The task-specific weights are stored in the *LoraBertSelfAttention* class and can be swapped out via a *switch_mode* function on the fly.

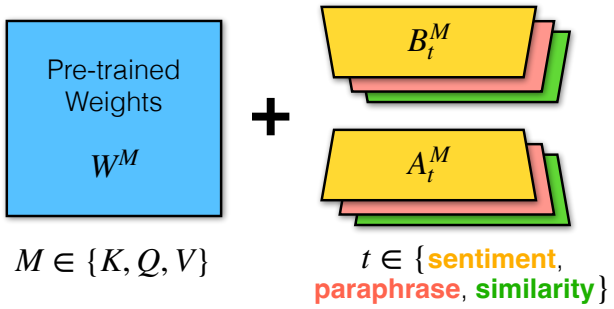

Figure 1: Schematic illustration of total weight matrix $W^M + \Delta W_t^M$. Colors represent task-specific fine-tune weights.

For the extension to DoRA,(Liu et al. (2024)) the *LoraBertSelfAttention* had an added boolean member *use_dora* that altered into

$$M = m\frac{(W^M + \Delta W_t^M)}{||W^M + \Delta W_t^M||_{2'}}X = m\frac{(W^M + B_t^M A_t^M)}{||W^M + B_t^M A_t^M||_{2'}}X \tag{3}$$

where $m = ||W^M||_{2'}$ and $||C||_{2'}$ is a row vector in which each element in index i corresponds to the 2-norm of the i-th column of matrix C. Similar to the LoRA implementation described above, the task-specific weights are stored within the extended *LoraBertSelfAttention* class and can be switched with the same *switch_mode* function on demand.

Data augmentation: In addition to employing parameter-efficient fine tuning methods, in this work I utilized two data-augmentation approaches to augment the data provided through the default project. Data augmentation, which is orthogonal to the parameter efficient training approaches above, was intended to further push the accuracy of the training models.

The first approach for data augmentation is back translation, in which the original sentence is translated to another language and translated back. For example, "Not all those who wander are lost" (a quote from Lord of the Rings) can be translated to "Ekki eru allir týndir sem villast" in Icelandic, which can be translated back to "Not everyone who goes astray is lost", which retains the same meaning. For the translation task, a transformer-based neural machine translation model was used. I have used Google's MADLAD-400-3B-MT model(Kudugunta et al. (2024)) to augment the sst and sts training sets by translating the English sentences to Korean and then back to English. The model was used without modifications using weights and architectures provided by Hugging Face.

The second approach for data augmentation was EDA.(Wei and Zou (2019)) EDA utilizes synonym replacement (e.g. "I am learning about deep learning **techniques**" to "I am learning about deep learning **methods**"), random insertions (e.g. "The weather is nice" to "The weather is tortoise nice"), swaps ("weather The is nice"), and deletions ("The weather nice").

Data augmentation was applied to the SST and SemEval STS data sets. Back translation was used to double the data set size and EDA was used augment each sentence by 9x. Augmentation was not applied to the Quora dataset which was already sufficiently large to avoid excessive use of resources during training.

5 Experiments

5.1 Data

Since the current project is the default 224N project, I have primarily utilized the training datasets provided through the project (SST, Quora, and SemEval STS). These train sets have been augmented

using the methods described in the previous section. During the data augmentation process, the ratios of label imbalance has been mitigated by not augmenting the Quora dataset. However with task-specific fine tuning, a separate set of fine-tuning weights are used for each task; the weights across tasks are entirely independent because they are swapped out for each task. Hence, the order of training for each task, as well as any kind of task-dependent data imbalance is not expected to produce negative effects.

5.2 Evaluation method

As previously discussed, the accuracies of each task from multi-task evaluations are reported for evaluation. Separately, memory and compute resources have been be tracked and reported to compare the strength of PEFT methods LoRA and DoRA. Furthermore, different LoRA/DoRA ranks r have been selected for comparison.

Memory (VRAM) usage and power consumption was measured by *nvidia-smi*. The compute usage is measured by throughput (iterations per seconds) during training. To eliminate external effects such as throttling which may result in unstable time measurements, GPU clock speeds will be appropriately locked based on hardware specifications. Current experiments were done on an NVIDIA RTX3090 GPU with graphics clock locked at 1695 MHz. For NVIDIA GPUs, SM clocks can be locked via the *nvidia-smi -lgc* command.

5.3 Experimental details

The baseline model for the default project is the default *MultitaskBERT* as provided by the assignment. The model uses a separate and single linear layer with dropout probability $p = 0.2$ for each of the sentiment analysis, paraphrase detection, and similarity detection task. Training for each task was done round-robin style; within each epoch, the model was trained with the SST, paraphrase, and STS dataset in order. Finally, the model was trained for 10 epochs each.

The LoRA and DoRA extensions to *MultitaskBERT* maintained the same architecture except for the fine-tuning weights. The availability of LoRA and DoRA, each with rank r, and the use of augmented data allows for a large number of combinations of model configurations. To avoid a combinatorial explosion of possibilities, the LoRA extension was tested with a single rank $r = 12$, DoRA with $r = 1, 4, 12, 24, 128$ without augmented data. Finally, DoRA with $r = 12$ and augmented data was used to test the efficacy of the larger dataset. The same training approach was taken during each model was trained for 10 epochs and round-robin style within each epoch. As mentioned in Section 5.1 above, the ordering of training does not affect training because each task is trained separately.

5.4 Results

The Dev set accuracies collected after the baseline, LoRA-MultiBERT and DoRA-MultiBERT with various fine tuning ranks r are shown in Table 1. Interestingly, when comparing the accuracies averaged across all three tasks, the DoRA-MultiBERT with rank $r = 12$ achieved the highest accuracy, including the LoRA variant or when augmented data was used.

While LoRA and DoRA outperforming the baseline MultitaskBERT is expected due to task-specific fine tuning capabilities, the results above indicate that data augmentation did not in fact improve the accuracy achieved by the model. Reasons underlying the accuracies achieved by each model in Table 1 are discussed in Section 6.

The test-set predictions for DoRA-MultiBERT with rank $r = 12$ was submitted to the Gradescope leaderboard. **The accuracies and correlations achieved were 0.524 (SST accuracy), 0.892 (Paraphrase accuracy), and 0.862 (STS correlation), with an overall test score 0.782.**

The measured resources (VRAM and power consumption) and throughput (it/s on SST task) results from training are summarized in Table 2. As expected, the default MultitaskBERT's memory footprint, VRAM usage is significantly heavier when training the full model. This is because the Adam optimizer requires additional floating point values within memory for parameters that can be updated. Our MultitaskBERT has roughly 109 million parameters (counted via PyTorch's *parameter.nelement()* function) and BERT uses roughly 18 bytes per parameter when training. Moreover, when comparing LoRA-MultitaskBERT against MultitaskBERT, the memory footprint is close to last-layer-only fine-tuning case while compute throughput is closer to that of the full-model.

Table 1: Training results for MultitaskBERT and LoRA/DoRA-MultitaskBERT

Trace	Fine-Tune Rank (r)	LR	SST Dev. Acc.	Paraphrase Dev. Acc.	STS Dev. Corr.	Dev. Avg.
MultitaskBERT (last layer)	0	1e-5	0.412	0.743	0.510	0.555
MultitaskBERT (full model)	768	1e-3	0.470	0.891	0.883	0.748
LoRA-MultiBERT	12	1e-4	0.519	0.890	0.835	0.748
DoRA-MultiBERT	1	1e-4	0.492	0.874	0.827	0.731
DoRA-MultiBERT	4	1e-4	0.510	0.889	0.842	0.747
DoRA-MultiBERT	**12**	1e-4	0.519	0.892	0.864	**0.758**
DoRA-MultiBERT	24	1e-4	0.506	0.892	0.845	0.748
DoRA-MultiBERT	128	1e-4	0.513	0.886	0.851	0.750
DoRA-MultBERT (Augmented Data)	12	1e-4	0.504	0.864	0.847	0.738

Table 2: Measured resource consumption and throughput for MultitaskBERT and LoRA/DoRA-MultitaskBERT

Trace	Fine-Tune Rank (r)	VRAM Usage	SST Thrpt.	Power Consump.
MultitaskBERT (last layer)	0	814 MB	115.5 it/s	320 W
MultitaskBERT (full model)	768	2622 MB	18.7 it/s	270 W
LoRA-MultiBERT	12	1222 MB	20.9 it/s	230 W
DoRA-MultiBERT	1	1218 MB	21.0 it/s	240 W
DoRA-MultiBERT	4	1222 MB	22.0 it/s	245 W
DoRA-MultiBERT	12	1232 MB	20.9 it/s	244 W
DoRA-MultiBERT	24	1248 MB	20.8 it/s	245 W
DoRA-MultiBERT	128	1384 MB	19.6 it/s	266 W
DoRA-MultiBERT (Augmented Data)	12	1232 MB	21.0 it/s	241 W

One notable factor in the result shown in Table 2 is the relative independence of fine-tune rank. The use of LoRA/DoRA results in VRAM usage and compute throughput between layer-layer tuning and full model tuning, but the rank r has small impacts on VRAM Usage and throughput. This result suggests that the choice of the value of r may be primarily dictated by the accuracy achieved, rather than resource constraints.

6 Analysis

6.1 Model Accuracy

The results shown in Table 1 generally indicate that LoRA and DoRA allow parameter-specific fine tuning that is able to achieve accuracies comparable to tuning the full model. One notable trend observable trend from Table 1 is the plateau of accuracy reached after $r \geq 12$. In other words, we see that increasing the fine-tune rank led to improved accuracy up to $r = 12$.

The diminishing return for higher r in the current experiment can be attributed to overfitting. Figure 2 shows the training and dev accuracies for SST and STS tasks measured and recorded during training for r =12, 24, and 128. Examining the solid lines (dev accuracies) indicate that the training quickly reaches terminal training accuracy within fluctuation in the first few epochs. However, the training accuracies continue to increase during training, and the training-validation gap increases significantly with larger fine-tune rank r, suggesting that the fine-tuning process is overfitting to the training data despite the use of dropout with probability $p = 0.2$. This observation also validates the claims from the original LoRA work,(Hu et al. (2021)) in which the weight updates during training appear to be

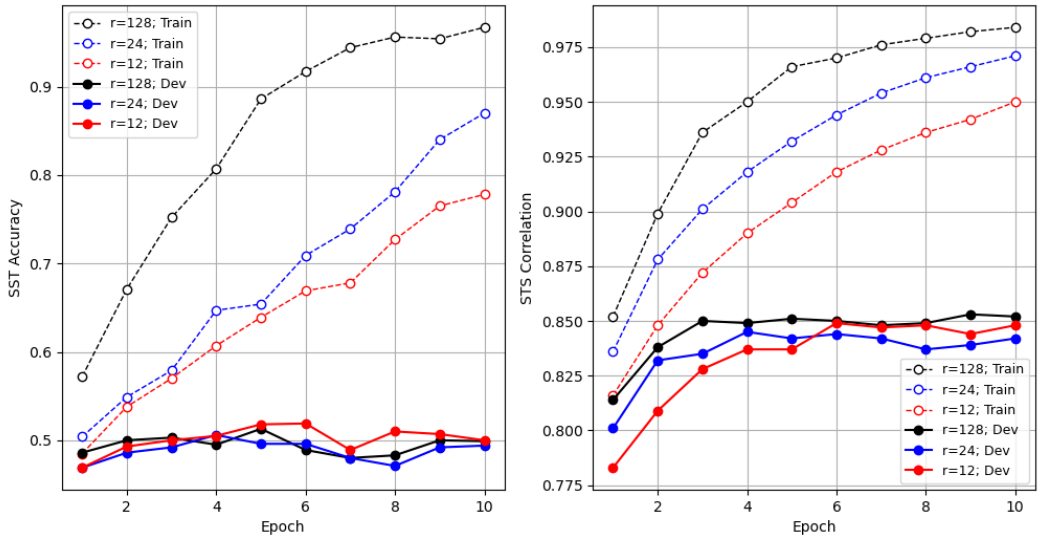

Figure 2: Training and dev SST Accuracy (left) and STS Correlation (right) during training for DoRA-MultiBert r=12, 24, and 128. Dashed line with open markers indicate training accuracies while solid lines with filled markers indicate dev accuracies.

low rank. The study in this work in deed shows that low-rank updates can sufficiently capture the necessary features of the training data that must be learned.

Several methods can be used to mitigate the overfitting observed in the training above. First, different hyperparameters can be explored. A single learning rate (1e-4) and dropout probability (0.2) was used for all experiments. Searching for different values, as well as a decay schedule, may improve generalization performance. Second, a different set of layers for generating predictions for each task could be used. In the current implementation, the multitask classifier uses a single layer after BERT's outputs are generated for each task to compute the prediction. While deeper neural networks are generally more susceptible to overfitting, including several layers that utilize dropout and other techniques may increase the general accuracy of predictions.

6.2 Augmented Data

Another result shown in Table 1 is the negative impact of utilizing augmented data, which is counterintuitive. the results indicate that all tasks suffered loss in dev accuracies. This may be due to the quality of translations in the augmented data. Inspection of the augmented data obtained via back translation using MADLAD-400-3B-MT(Kudugunta et al. (2024)) show translations of mixed quality. In some cases, the translation was acceptable; for example:

- Original Sentence: Good fun , good action , good acting , good dialogue , good pace , good cinematography . (Sentiment 4)
- Back translation: Fun, good action, good acting, good conversation, good speed, good shooting.

are two sentences that sound sufficiently similar. In other cases, there were sentences that have completely lost their meaning

- Original Sentence: An absurdist spider web . (Sentiment 1)
- Back translation: Improves the network speed of network players.

The overall quality of back translated sentences depends on the model and languages being translated. While the MADLAD-400 family of models include 7B and 10B parameter variants, the 3B model was used to rapidly generate translations. Using a larger variant will likely improve translation qualities. Moreover, the choice of backtranslation may impact the quality of translation. Here,

English sentences were translated to Korean and back. Choosing a language that is better handled by the neural translation model may lead to better augmented data.

Inspection of sentences augmented via EDA (Wei and Zou (2019)) show mixed quality as well. In the following example,

- Original Sentence: The overall fabric is hypnotic , and Mr. Mattei fosters moments of spontaneous intimacy . (Sentiment 4)
- EDA 1: the overall fabric is hypnotic and mr mattei foster moments cloth of spontaneous intimacy. (random insertion)
- EDA 2: the overall fabric hypnotic and mr mattei fosters of sponatneous intimacy (random deletion)
- EDA 3: the overall is hypnotic and mister mattei fosters moments of spontaneous intimacy. (random deletion)
- EDA 4: the overall stephen foster fabric is hypnotic and mr mattei foster moments of sponatenous intimacy (random insertion, switch ordering)

the original sentence conveys a strongly positive sentiment. While the augmented sentences all appear to maintain a positive review, not all augmented sentences seem carry equally strong sentiments as the original sentence. As such, the augmented dataset may have caused a deleterious effect on the sentence sentiment evaluation task. This shows that (1) the quality of augmented data should always be examined prior to use; and (2) when using augmented data for training, one may want to carefully evaluate whether a data augmentation technique is suitable for generating data directed for the given task.

7 Conclusion

In this CS224N Default Project, I have investigated the efficacy of parameter efficient fine tuning methods LoRA and DoRA to multitask-BERT. Starting from a default BERT implementation and pre-trained weights, a multitask-BERT model was implemented using a task-dependent last layer. Then, the low-rank fine tuning method was implemented by additionally story the low-rank matrices B and A. To impart task-specificity, the multitask-BERT model was initialized with separate low rank matrices B and A for each task, and a *switch_mode(mode)* function was implemented to replace the currently active fine-tuning weights on the fly.

Using the Multititask-BERT as a baseline, experiments were conducted by training the baseline model and LoRA/DoRA extensions. During the investigation, augmented datasets were also employed to study the usefulness of added synthetically augmented data. The results show that the LoRA(Hu et al. (2021)) authors claim of "efficient training with no accuracy loss" is indeed achievable, even with some naive hyperparameter tuning. However, the characterization of memory and compute resources showed that the training was memory-efficient but not *compute efficient*; VRAM footprint was small but training took nearly as long as training the full model. Nevertheless, in the current age of explosive growth in model size, memory efficiency may be far more valuable than training throughput; parameter efficient fine-tuning techniques such as LoRA and DoRA do indeed enable training of large models in hardware accelerators with limited VRAM capacity.

Upon experimenting with several values of rank r, this study shows that weight updates to a transformer-based model do indeed have a tendency for low rank. This is exemplified by the result in which increasing the rank beyond a certain yet low rank value of $r = 12$ showed no meaningful improvement in validation (dev) set accuracy. Finally, data augmentation techniques were applied to increase the amount of data used in training, with hopes to increase training and dev set accuracy. Yet, the augmented data was shown to yield no avail and in fact rather cause slight loss in accuracy.

One key limitation of the experiments in this study is the lack of sufficient hyperparameter tuning. A naive learning rate of 1e-4 was selected without further experimentation for all parameter-efficient fine tuning jobs. Moreover, a single dropout probability (0.2) was used. Given sufficient resources and time, one may perform a basic hyperparameter tuning to achieve higher accuracy. Another avenue for improvement is modification of the architecture of layers used after BERT. In the current implementation, each task is given a single layer to generate a prediction. A larger number of layers

or architectures such as embeddings may be utilize to further make use of predictions returned by BERT.

8 Ethics Statement

The current project expands on the default project, utilizes pre-trained BERT and data from the Stanford Sentiment Treeband dataset, CFIMDB dataset, Quora dataset, and SemEval STS Benchmark dataset. Additionally, a trained neural translation model, MADLAD-400 was used to generate part of the augmented data (EDA does not involve machine learning). The tasks at hand are sentiment analysis, paraphrase detection, and semantic similarity identification, which directly evaluate sentences.

The pre-trained weights, datasets, and data augmentation tools may contain explicit or implicit biases in evaluating the sentences at hand. Specifically, these biases may include unjustified prejudices against certain ideas or people of specific groups such as gender, race, or even occupations. The ramifications of such unfair biases can lead to incorrect classifications. As such, the tasks involved in this work do not face the same risks as certain tasks such as sentence generations; the model cannot generate hateful speech, unsupported claims, or unsafe advice. Nevertheless, implicit biases may be present. As an attempt to mitigate introduction of biases in the augmentation step, I have selected a neural translation model developed by a well-establish entity (Google) and an algorithmic approach (EDA) for data augmentation, with the a that the publicly released MADLAD-400 model should have been trained with "clean" data.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. 2022. Parameter-efficient fine-tuning for vision transformers. *arXiv preprint arXiv:2203.16329*, 3.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Sneha Kudugunta, Isaac Caswell, Biao Zhang, Xavier Garcia, Derrick Xin, Aditya Kusupati, Romi Stella, Ankur Bapna, and Orhan Firat. 2024. Madlad-400: A multilingual and document-level large audited dataset. *Advances in Neural Information Processing Systems*, 36.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

NVIDIA. 2017. Nvidia tesla v100 gpu architecture. Technical report.

NVIDIA. 2020. Nvidia a100 tensor core gpu architecture. Technical report.

NVIDIA. 2022. Nvidia h100 tensor core gpu architecture. Technical report.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. 2023. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*.