Parameter Efficient Fine-Tuning for Multi-Task BERT

Stanford CS224N {Default} Project

Zhen Wu
Computer Science
Stanford University
zhenwu@stanford.edu

Genghan Zhang
Computer Science
Stanford University
zgh23@stanford.edu

Alexa Hu
ICME
Stanford University
xuningh@stanford.edu

Abstract

Fine-tuning pre-trained language models such as BERT for downstream tasks is essential but often computationally expensive. Parameter-efficient fine-tuning (PEFT) methods aim to optimize performance while minimizing resource utilization. In this project, we investigate PEFT techniques for minBERT across various downstream tasks. We apply the LoRA method to reduce parameters and computational complexity. Furthermore, we integrate PEFT with multi-task learning paradigms, incorporating techniques such as the DoRA extension and orthogonal adaptation. Our approach aims to achieve efficient fine-tuning, providing a balanced trade-off between performance and resource utilization.

1 Key Information to include

- TA mentor: Johnny Chang; External collaborators (if no, indicate "No"): No; External mentor (if no, indicate "No"): No; Sharing project (if no, indicate "No"): No
- Contributions:
  - Zhen Wu: Implement LoRA for multi-task setting and run experiments. Write Section 4.2, Section 5, and Section 6.2.
  - Genghan Zhang: Implement the last layer, sequential training, LoRA and DoRA for single task. Write Section 4.1, Section 6.1, and Section 7.
  - Alexa Hu: Implement minBERT handout section 4.2 and 4.4. Write Abstract, Section 2, Section 3, and Section 8 of report.

2 Introduction

The fine-tuning of pre-trained large language models for downstream tasks has become a cornerstone of modern natural language processing. These models are initially trained on vast corpora, requiring adaptations to perform specific tasks effectively. However, this fine-tuning process often demands substantial computational time and resources. To address this issue, the Parameter-Efficient Fine-Tuning (PEFT) technique has been developed to retain the performance benefits of full fine-tuning while significantly reducing the resource requirements.

In this paper, we delve into the application of PEFT techniques to minBERT across a variety of downstream tasks. Our primary focus is on the Low-Rank Adaptation (LoRA) (Hu et al., 2021), thereby reducing the number of parameters that need to be fine-tuned. This approach not only decreases the computational complexity but also maintains the model's performance on specific tasks.

Building on LoRA, we further explore the integration of PEFT with multi-task learning paradigms, which allows a model to be trained on multiple tasks simultaneously, leveraging shared information across tasks to improve overall performance and efficiency. We incorporate Weight-Decomposed Low-Rank Adaptation (DoRA) (Liu et al., 2024), which introduces regularization to adjust the rank of the adaptation matrices dynamically. Additionally, we apply orthogonal adaptation (Po et al., 2023)

to LoRA. We generate a shared orthogonal basis and sample LoRA weights from the basis to enforce orthogonality, minimizing interference between tasks while preserving the model's expressiveness.

Our evaluation encompasses diverse downstream tasks, specifically sentiment analysis, paraphrase detection, and semantic textual similarity. Through experiments, we demonstrate the effectiveness of our proposed methods in achieving parameter efficiency and computational savings while maintaining high levels of task performance.

3 Related Work

Pretraining a large model on general language tasks has proven beneficial for enhancing the accuracy and performance of various natural language processing tasks (Radford et al., 2018). However, adapting large-scale pre-trained language models (PLMs), especially those with billions of parameters is challenging. Full fine-tuning of all parameters in large models is expensive and impractical, particularly when storing and managing a separate model for each task. Parameter-efficient multitask fine-tuning methods aim to address this by training a single model for various downstream tasks, updating only a small subset of parameters. Specifically, PEFT methods(Xu et al., 2023) can maintain the performance of PLMs while significantly reducing the computational resources required for fine-tuning. PEFT techniques are being widely adopted in various models, such as diffusion models Zhang et al. (2023), vision-language models Dai et al. (2024), and language models Zhao et al. (2024).

Instead of updating all model parameters, Hu et al. (2021) proposes LoRA, which freezes the pre-trained model weights and injects trainable low-rank decomposition matrices into each layer of the transformer architecture. While LoRA efficiently minimizes the number of trainable parameters by injecting low-rank matrices into pre-trained models, it often exhibits an accuracy gap. DoRA (Liu et al., 2024) mitigates this gap by decomposing the pre-trained weight into magnitude and direction. It employs LoRA specifically for updating the directional component, thus reducing the number of trainable parameters while maintaining the learning capacity and stability.

In multi-task learning, tasks are often interrelated and can negatively impact each other's performance if not managed properly. Orthogonal adaptation merges individually fine-tuned models without compromising fidelity or incurring additional computational costs, originally applied for modular customization of text-to-image diffusion models (Po et al., 2023). During fine-tuning, only one component of the weight residuals is optimized, while the other remains fixed, leveraging a shared orthogonal basis to ensure approximate orthogonality.

In this project, we investigate how various factors influence LoRA's performance and provide valuable insights into its application in multi-task learning. We utilize DoRA to improve upon LoRA's efficiency in managing downstream tasks with adding neglible inference latency. Additionally, we explore the impact of orthogonal adaptation on LoRA and its effectiveness across various downstream tasks.

4 Methods

In this section, we introduce the full-model fine-tuning and parameter-efficient fine-tuning techniques. We carefully design the last layer and the training recipe to fully utilize the training data and achieve better generalization.

4.1 Full-Model Fine-Tuning

The last layer is the same for full-model and parameter-efficient fine-tuning. The parameter-efficient fine-tuning has a more complex training recipe that includes model ensemble.

4.1.1 Last Layer Design

We first introduce the last prediction layer $\mathcal{L}_{W,b}$ for the para and sts tasks. Both tasks take in two sentences x_1 and $x_2 \in \mathbf{R}^F$, and output one scalar $y \in \mathbf{R}$. Denote a Bert base model as $\mathcal{B}$ that outputs the [CLS] token representation from the last transformer block. Our first implementation is

$$\mathcal{L}_{W,b} \circ \mathcal{B}(x_1, x_2) = [\mathcal{B}(x_1); \mathcal{B}(x_2)]W + b \tag{1}$$

where $W \in \mathbf{R}^{2F \times 1}$ and $b \in \mathbf{R}$. Basically, we forward the same Bert model twice, concatenate the [CLS] token representations, and apply an affine transformation. After that, we generalize $\mathcal{L}_{W,b}$ to a projection function $\mathcal{P}_\theta$ and an aggregation function $\mathcal{A}$. Therefore, we propose the second and third implementations:

$$\mathcal{A} \circ \mathcal{P}_\theta \circ \mathcal{B}(x_1, x_2) = \mathcal{A}(\mathcal{B}(x_1)W_1 + b_1, \mathcal{B}(x_2)W_2 + b_2) \tag{2}$$

where $\theta = W_1, W_2 \in \mathbf{R}^{F \times d}$ and $b_1, b_2 \in \mathbf{R}^d$. However, Equation (1) and (2) do not utilize the attention mechanism to aggregate the information of two input sentences. Instead, x_1 and x_2 are independently encoded. Although the loss propagates the downstream task information, Equation (1) leaves some representation ability on the table. Therefore, we concatenate the two sentences for $\mathcal{B}$.

$$\mathcal{A} \circ \mathcal{P}_\theta \circ \mathcal{B}(x_1, x_2) = \mathcal{A}(\mathcal{B}([x_1; x_2])W_1 + b_1, \mathcal{B}([x_1; x_2])W_2 + b_2) \tag{3}$$

We design $\mathcal{A}$ for specific tasks: inner-product for paraphrase prediction and scaled cosine similarity for similarity prediction, as shown in Equation (4) and (5), respectively. We used Equation (5) to scale the $[-1, 1]$ cosine similarity to $[0, 5]$ sentence similarity and set $\alpha = 1$, $\beta = 2.5$.

$$\mathcal{A}(\vec{v_1}, \vec{v_2}) = \vec{v_1} \cdot \vec{v_2} \tag{4}$$

$$\mathcal{A}(\vec{v_1}, \vec{v_2}) = (\frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\| \, \|\vec{v_2}\|} + \alpha) * \beta \tag{5}$$

4.1.2 Sequential Training

We use sequential training for full-model fine-tuning, where we train a single model sequentially on multiple datasets. During this process, the outer loop iterates over the number of epochs, the middle loop iterates over each downstream dataset, and the inner loop processes all batches of data from each dataset. We keep the optimizer states across datasets.

4.2 Parameter Efficient Fine-Tuning

While fine-tuning the full model yields good results, it is computationally expensive. Therefore, we explore PEFT methods, aiming to optimize performance while minimizing resource utilization. Initially, we implemented popular PEFT methods, including LoRA (Hu et al., 2021) and DoRA (Liu et al., 2024), ourselves and conducted experiments on minBERT. We also experimented with various approaches to extend LoRA to multi-task settings.

4.2.1 Vanilla LoRA

The idea behind LoRA is that during fine-tuning, the updates to the weight matrices are essentially low-rank. Thus, given a pre-trained model and its weight matrix $W_0 \in \mathbb{R}^{d \times k}$, instead of updating W_0 directly, LoRA re-parameterizes the weights by $W_0 + AB$, where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$. Here, r is much smaller than d and k, which significantly reduces the number of parameters that need to be learned during fine-tuning, thus making the process more efficient.

In the original LoRA paper, the authors kept all parameters of the pre-trained model frozen and applied LoRA only to the query and value matrices in all attention layers. To further enhance performance, subsequent work has expanded the application of LoRA to include all linear layers and set the bias vectors in the pre-trained model as trainable. We experimented with these various settings, which will be discussed in detail in the next section.

4.2.2 Regularized LoRA: DoRA

LoRA may underperform compared to fine-tuning the full model. To address this, DoRA has been proposed. During inference, DoRA modifies the weight matrix W to be $\mathbf{m}\frac{W_0+AB}{\|W_0+AB\|}$, where $\mathbf{m} \in \mathbb{R}^{1 \times F}$ is a learnable scaling factor, and W_0, A, and B are the same as in LoRA. Intuitively, this technique attempts to decouple updates to the magnitude and direction of weights, potentially improving the stability and performance of the model.

4.2.3 Multi-Task LoRA

We also explore methods to extend LoRA to a multi-task setting. We experiment with two approaches. First, we train a single model sequentially on multiple datasets. Second, we train separate models on different datasets and then ensemble them together.

Sequential Training Similar to fine-tuning the full model as described in Section 4.1.2, our first trial involves training LoRA sequentially on multiple datasets, and we can change the order of datasets for different training dynamics.

Model Ensemble In addition to training a single model on multiple downstream tasks, we also trained separate models on different datasets and then ensembled them together. Specifically, we experimented with the following methods:

1. Ensemble of Experts (EoE): We trained a set of LoRA parameters for each downstream task. During inference, we used the corresponding set of parameters and classifier head for predictions. Specifically, for each task i, we used the corresponding LoRA parameters $\Delta\theta_i$ to adjust the base model parameters θ, resulting in the task-specific parameters $\theta_i = \theta + \Delta\theta_i$, and we use θ_i to inference task i. We call this method an ensemble of experts because each set of LoRA parameters acts like an expert on each task and they are ensembled at inference time.

2. Federated Averaging (FedAvg): As suggested in (McMahan et al., 2017), we trained a set of LoRA parameters for each downstream task, and then we took a weighted average of each set of parameters. Given a set of LoRA parameters $\Delta\theta_i$ optimized on downstream task i, the resulting merged model is given by

$$\theta_{\text{merged}} = \theta + \sum_i \lambda_i \Delta\theta_i, \tag{6}$$

where θ represents the pre-trained parameters, and λ_i is a scalar representing the relative strength of each task.

3. Orthogonal Adaptation (OrthoAda): The linear combination of weights in FedAvg can lead to performance loss due to interference between the learned weight residuals. Intuitively, if we can make the weight residuals orthogonal to each other, we can mitigate this loss. To achieve this, note that weight residuals for task i in LoRA can be represented in $\Delta\theta_i = A_i B_i^T$, if we can make sure all $B_i^T B_j = 0, i \neq j$, then we will have $\Delta\theta_i \Delta\theta_j^T = 0$. We can accomplish this by first generating a large orthogonal matrix and then selecting different columns from it to form B_i and B_j. This method has been proven effective in fine-tuning large text-to-image models, as suggested in (Po et al., 2023). Therefore, we experiment with this approach in our multi-task setting to evaluate its effectiveness.

5 Experiments

5.1 Data

We used the datasets specified in the handout to fine-tune our models for three downstream tasks.

- **Sentiment Analysis**: We utilized the Stanford Sentiment Treebank (SST), which includes 11,855 sentences derived from movie reviews. These sentences are annotated with ratings that range from negative to positive.
- **Paraphrase Detection**: For this task, we used the Quora Dataset (Para), consisting of 400,000 question pairs. Each pair is labeled to indicate whether they are paraphrases of one another.
- **Semantic Textual Similarity**: We employed the SemEval STS Benchmark (STS) dataset for this task. It contains 8,628 sentence pairs, each rated on a scale from 0 (unrelated) to 5 (equivalent in meaning).

5.2 Evaluation method

As described in the default project handout, for Sentiment Analysis and Paraphrase Detection, we evaluated performance using the mean accuracy on class labels. For Semantic Textual Similarity, we assessed performance by calculating the Pearson correlation between the true similarity values and the predicted similarity values.

5.3 Experimental details

For both full-model fine-tuning and PEFT, We use lr=1e-5, dropout=0.3, batch=32, epochs=10. In Table 1, we use sequential training (Section 4.1.2) set $d = 64$ for Equation (2) and (3). We test

different r and trainable parameter settings for PEFT on single task. Finally, for all PEFT experiments in the multi-task setting, we apply sequential training and model ensemble (Section 4.2.3) and adopt $r = 64$, apply LoRA to all linear layers, setting the bias as not trainable. Figure 1 illustrates all the settings we have experimented with.

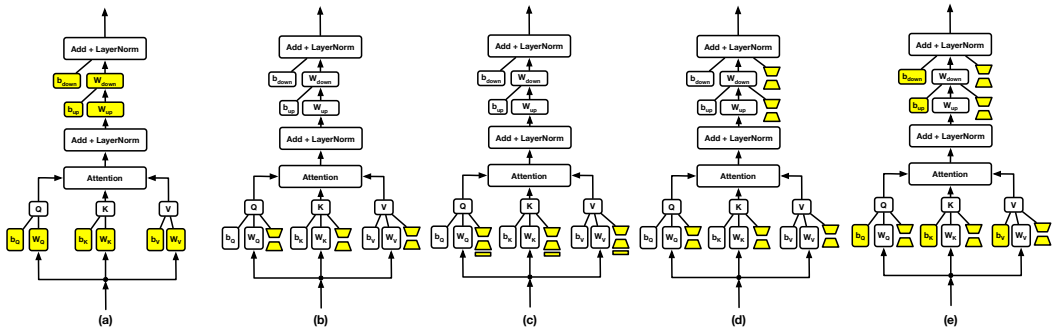

Figure 1: A single transformer layer that composes of attention and feed-forward network (FFN). The yellow blocks are updated during finetuning. (a) Full parameter used for Table 1; (b) LoRA (query & value & key) where the trapezoids represent low-rank weights; (c) DoRA (query & value & key) where the small rectangles under the trapezoids represent the norm $\mathbf{m}$; (d) LoRA (all linear); (e) LoRA (all linear & bias)

5.4 Results

In Table 1, we report the accuracy of full-model fine-tuning on each downstream task as well as the overall score. This model is the one we submitted to the leaderboard. We discuss our experiments with different designs of the last layer's structure and sequential training order in Section 6.1.

Additionally, in Table 2, we report the PEFT results on the single task setting. We experimented with the following factors to see how they affect model performance:

- Different LoRA dimensions r,
- Regularized LoRA: DoRA,
- Whether to apply LoRA only to the attention layer or to all linear layers.

We report the PEFT results in the multi-task setting in Table 3. Here, we first provide the results for different orders of sequential training. We also show the results of the three methods of model ensembling. A detailed analysis is provided in Section 6.2.

Table 1: Leaderboard scores. The results are obtained by full-model fine-tuning.

Task	SST	Para	STS	Overall
Dev	0.519	0.897	0.864	0.783
Test	0.501	0.897	0.874	0.778

6 Analysis

6.1 Full-Model Fine-Tuning

We first analyze the design of the last layer. For the three designs we proposed, described in Equation (1), (2), (3) respectively, we observe the following results. As shown in Figure 2, Method 3 significantly improves the model's performance on STS and also benefits the other two tasks. Therefore, we choose Method 3 as the last layer for the later experiments.

We also experiment with the order of the sequential training. No single order consistently dominates either on a specific dataset or on the overall score. The dataset order also affects the model perfor-

Table 2: PEFT single task scores. The best results are in bold.

Method	LoRA Dim r	SST	Para	STS	Overall	Parameters updated
LoRA (query & value & key)	2	0.421	0.838	0.664	0.697	0.10%
LoRA (query & value & key)	4	0.425	0.848	0.699	0.707	0.20%
LoRA (query & value & key)	16	0.464	0.862	0.761	0.736	0.80%
LoRA (query & value & key)	64	0.468	0.878	0.804	0.749	3.13%
DoRA (query & value & key)	64	0.480	0.876	0.802	0.752	3.15%
LoRA (all linear)	64	0.490	0.886	**0.823**	0.762	8.91%
LoRA (all linear & bias)	64	0.490	0.886	**0.823**	0.762	8.96%
LoRA (all linear & bias)	128	**0.493**	**0.893**	**0.823**	**0.766**	16.45%

Table 3: PEFT multi-task scores. The best results are in bold.

	Order / Method	SST	Para	STS	Overall	Parameters updated
Sequential Training	SST-Para-STS	0.473	0.882	0.826	0.756	8.91%
	SST-STS-Para	0.429	0.886	0.829	0.743	8.91%
	Para-SST-STS	0.483	0.881	0.830	0.760	8.91%
	Para-STS-SST	0.471	0.883	0.834	0.757	8.91%
	STS-SST-Para	0.482	0.887	**0.840**	**0.763**	8.91%
	STS-Para-SST	0.478	**0.889**	0.826	0.760	8.91%
Model Ensemble	EoE	**0.490**	**0.886**	**0.823**	**0.762**	22.8%
	AvgFed	0.324	0.637	0.592	0.586	8.91%
	OrthoAda	0.233	0.657	0.678	0.576	8.91%

mance in different ways. For SST, the trends for Method 2 and 3 are similar, while for STS, the trends for Method 1 and 2 are similar. However, the overall score is robust on the dataset order.

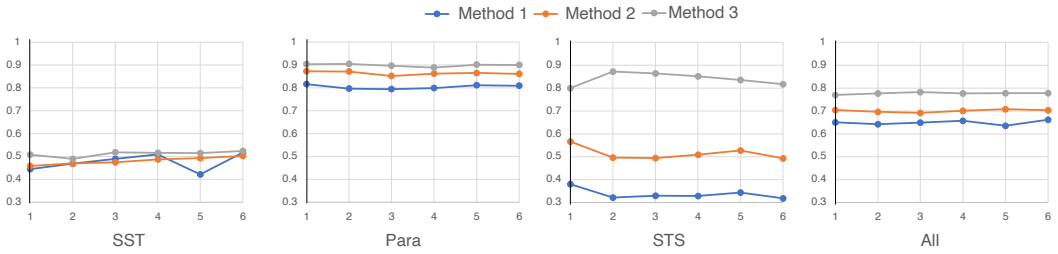

Figure 2: Results of Method 1, 2, 3 across 6 order of datasets (SST-STS-Para, SST-Para-STS, Para-SST-STS, Para-STS-SST, STS-SST-Para, STS-Para-SST). The horizontal axis represents these dataset orders.

6.2 Parameter Efficient Fine-Tuning

We first experimented with LoRA performance in the single-task setting, with the results shown in Table 2. We initially investigated the impact of different LoRA dimensions r on performance. As shown in the first four rows of Table 2, performance improves as r increases, indicating that **more trainable parameters lead to better performance**. Notably, by comparing the third and first rows of Table 2, we can see that LoRA achieves 94% of the full-model fine-tuning score while updating only 0.80% of the parameters, demonstrating the effectiveness of PEFT.

DoRA is better than LoRA on SST. We compared LoRA with DoRA. As shown in the fourth and fifth rows of Table 2, DoRA significantly improved performance on the SST dataset, although it did not show substantial improvements on the other two datasets. Since DoRA serves as weight normalization over the updated parameters, it might perform better than LoRA when the updated weights' norm changes abruptly. Therefore, DoRA can help on the SST because new weights added by LoRA might be so large that leads the model to a worse local optimal point.

We further compared different LoRA settings, specifically whether to apply LoRA only to the attention layer or to all linear layers, and whether to fine-tune the bias in the linear layers. These comparisons are illustrated in the fourth and last three rows of Table 2.

- **Applying LoRA to linear layers in FFN can further improve the model**. By comparing the fourth row with the third row from the bottom, we found that applying LoRA to all linear layers significantly improved performance, though it also increased the number of parameters being updated. This trend aligns with our first observation that more trainable parameters can lead to better performance.
- **Updating bias at the same time does not improve the model**. Comparing the second and third rows from the bottom, we observed that fine-tuning the bias had minimal impact, likely because the bias constitutes a very small portion of the total parameters.
- **Increasing the inner-dimension can also improve LoRA on FFN similar to the attention**. Finally, comparing the second row from the bottom with the last row, we found that increasing the LoRA dimension r further improved performance. This improvement was mainly achieved by increasing accuracy on SST and Para, while STS performance remained stable, indicating that increasing the number of trainable parameters no longer enhanced STS performance.

No single order of sequential training consistently dominates in the PEFT multi-task. We then show the sequential training results using LoRA in the upper half of Table 3. Similar to sequential training on full-model fine-tuning, no single order consistently dominates either on a specific dataset or in terms of the overall score. That is because the sequential training order changes the training dynamics and might serve as beneficial regularization for a task.

EoE performs better than other model ensemble methods. We also present the results of model ensemble methods in the lower half of Table 3. The EoE method, which combines the LoRA weights trained separately on each task, achieves relatively high accuracy but involves updating a large number of parameters. In contrast, AvgFed and OrthoAda use a weighted sum of the LoRA weights from single tasks as the new weights. While this approach requires fewer parameters to be updated, the performance is not as strong. AvgFed was designed for federated learning where parameters of different models are merged multiple times. However, we only merge the weights once in our experiments because we do not have enough training data, although McMahan et al. (2017) claims that AvgFed can perform well on imbalanced training data similar to our setting. OrthoAda was used for isolating weights for different models fine-tuned independently for individual concepts. It performs worse than EoE because the 3 tasks are not independent, and merging the weights orthogonally deteriorates the model's performance on other tasks.

7 Conclusion

We first explore different designs of the last layer and different ways of composing training datasets in multitask fine-tuning. We find that extra aggregation can benefit the model, and achieve 0.778 overall score on the Test Leardarboard. Then, we investigate the parameter efficient fine-tuning techniques (PEFT) on single-task and multi-task. For a single task, we examine two methods, LoRA and DoRA, from existing literature. For multi-task, we explore another orthogonal direction: different model ensemble methods. We find that the more parameters are updated, the higher quality the model has. Our work is limited to a small scale. Therefore, it will be interesting to explore the PEFT design space of larger models in the future.

8 Ethics Statement

Our project presents several ethical challenges and potential societal risks, including inheriting and amplifying biases presented in the training data, which leads to discriminatory outcomes. Privacy and security are also the major concerns, as fine-tuning sensitive data can risk unauthorized access and misuse. Additionally, the increased accessibility of advanced language models raises the potential for misuse in generating misinformation or malicious content. Despite the reduced computational requirements of LoRA, the environmental impact remains significant due to energy consumption. To mitigate these risks, we propose implementing bias detection, adopting privacy-preserving methods

such as differential privacy, and establishing ethical use policies. We also advocate for optimizing model architectures to reduce energy consumption and promote the use of renewable energy resources. By addressing these ethical challenges, we aim to ensure our work contributes positively to society while minimizing potential harm.

References

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2024. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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*.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Ryan Po, Guandao Yang, Kfir Aberman, and Gordon Wetzstein. 2023. Orthogonal adaptation for modular customization of diffusion models. *arXiv preprint arXiv:2312.02432*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pretraining.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.