KD-LoRA: A Hybrid Approach to Efficient Fine-Tuning with LoRA and Knowledge Distillation

Rambod Azimi[1] **Rishav Rishav**[1,3] **Marek Teichmann**[2] **Samira Ebrahimi Kahou**[1,3,4]

Abstract

Large language models (LLMs) have demonstrated remarkable performance across various downstream tasks. However, the high computational and memory requirements of LLMs are a major bottleneck. To address this, parameter-efficient fine-tuning (PEFT) methods such as low-rank adaptation (LoRA) have been proposed to reduce computational costs while ensuring minimal loss in performance. Additionally, knowledge distillation (KD) has been a popular choice for obtaining compact student models from teacher models. In this work, we present KD-LoRA, a novel fine-tuning method that combines LoRA with KD. Our results demonstrate that KD-LoRA achieves performance comparable to full fine-tuning (FFT) and LoRA while significantly reducing resource requirements. Specifically, KD-LoRA retains 98% of LoRA's performance on the GLUE benchmark, while being 40% more compact. Additionally, KD-LoRA reduces GPU memory usage by 30% compared to LoRA, while decreasing inference time by 30% compared to both FFT and LoRA. We evaluate KD-LoRA across three encoder-only models: BERT, RoBERTa, and DeBERTaV3. Code is available at https://github.com/rambodazimi/KD-LoRA.

1 Introduction

With advancements in transformer (Vaswani et al., 2017) architectures and hardware capabilities, including GPUs and distributed training, researchers have been able to develop LLMs with billions of parameters (Li et al., 2020; Narayanan et al., 2021; Dash et al., 2023), such as LLaMA 3.1 (Dubey et al., 2024) which boasts up to 405 billion parameters. These models, trained on trillions of tokens, exhibit remarkable capabilities across various downstream tasks (Brown et al., 2020; Zhuang et al., 2021; Wei et al., 2022). However, fine-tuning these models requires substantial energy and memory demands (Samsi et al., 2023). Furthermore, in recent years, the growth in the number of parameters in LLMs has significantly outpaced the advancements in available GPU memory (Lialin et al., 2023), amplifying the challenges of managing memory during fine-tuning (Singh et al., 2024; Kim et al., 2024; Dong et al., 2024).

To address these challenges, PEFT techniques (Houlsby et al., 2019) have emerged as effective solutions, which fine-tune a small subset of parameters while keeping the majority fixed. As shown in Figure 1, one popular PEFT technique, LoRA (Hu et al., 2022), and its variants (Zhang et al., 2023a; Zi et al., 2024; Ren et al., 2024; Zhao et al., 2024a; Liu et al., 2024) reduce the number of trainable parameters by introducing small, trainable rank decomposition matrices, maintaining performance as FFT across many tasks (Dettmers et al., 2023). For example, DoRA (Liu et al., 2024) enhances LoRA by decomposing pre-trained weights into magnitude and direction, applying LoRA to directional updates for reduced trainable parameters. Similarly, AdaLoRA (Zhang et al., 2023a) improves LoRA by dynamically allocating parameters based on their importance, optimizing efficiency and performance, particularly under tight budget constraints.

[1]Mila – Quebec AI Institute, [2]CM Labs Simulations Inc, [3]University of Calgary, [4]Canada CIFAR AI Chair

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

However, the effectiveness of PEFT methods varies across LLMs based on several factors such as model architecture and task type (Pu et al., 2023; Lee et al., 2023). Additionally, LoRA still requires substantial memory, as it does not reduce the activation memory cost compared to FFT (Chen et al., 2016; Zhang et al., 2023b; Zhao et al., 2024b). For example, a GPT-like model with 1.5 billion parameters, a sequence length of 1K, and a batch size of 32 requires approximately 60 GB of GPU memory (Rajbhandari et al., 2020). Moreover, LoRA does not improve inference time, as the full model still needs to be processed during inference (Liao et al., 2023; Gu et al., 2024a).

KD (Hinton et al., 2015) has become another prominent way to make the training and inference less memory-intensive by transferring capabilities of larger teacher models, such as GPT-4 (OpenAI et al., 2024), Gemini (Anil et al., 2024), and LLaMA (Dubey et al., 2024) to smaller student models without greatly compromising performance (Gu et al., 2024b; Xu et al., 2024).

KD has for instance been used to distill the BERT model into TinyBERT (Jiao et al., 2020) that has only 14.5 million parameters without significant performance loss. The performance of the distilled 11B parameter T5 model (Hsieh et al., 2023) has been shown to even surpass that of the much larger 540B parameter PaLM teacher model.

In this paper, we introduce KD-LoRA, a novel fine-tuning method that integrates LoRA into the KD framework to achieve competitive performance with reduced computational costs, making it ideal for deployment in resource-limited environments. We accomplish this by incorporating LoRA matrices into the student model and then applying the distillation process while updating the LoRA matrices of the student model. By combining KD with LoRA, we leverage the strengths of both methods: LoRA's efficiency in reducing trainable parameters and KD's ability to effectively transfer knowledge to more compact student models, resulting in reduced model size and shorter inference time.

We evaluate the effectiveness of KD-LoRA in comparison to FFT and LoRA across three encoder-only LLMs: BERT (Devlin et al., 2019), RoBERTa (Zhuang et al., 2021), and DeBERTaV3 (He et al., 2023). For the KD component, we select a smaller student model from the same family for each LLM. For each GLUE benchmark task (Wang et al., 2019), we explore various hyperparameter configurations and PEFT settings, utilizing NVIDIA A100 GPUs. The median performance is reported based on the top 6 configurations. Our comprehensive experiments on the GLUE benchmark reveal that KD-LoRA offers several advantages:

- KD-LoRA achieves about 97% of FFT's performance while updating significantly fewer parameters. For instance, FFT fine-tunes all 110M parameters of the BERT-base model, whereas KD-LoRA fine-tunes only 1.2M parameters with a rank of 8.
- KD-LoRA achieves about 98% of LoRA's performance while incorporating knowledge from a larger teacher model and using fewer trainable parameters due to the more compact student model. For example, LoRA fine-tunes 2.9M parameters of the RoBERTa-base model with a rank of 8, whereas KD-LoRA fine-tunes only 1.5M parameters with the same rank.
- KD-LoRA is 40% more compact than both FFT and LoRA by utilizing a smaller student model. This approach also reduces GPU memory usage by approximately 75% compared to FFT and 30% compared to LoRA during training.
- KD-LoRA reduces inference time by approximately 30% while maintaining the same convergence speed as both FFT and LoRA.

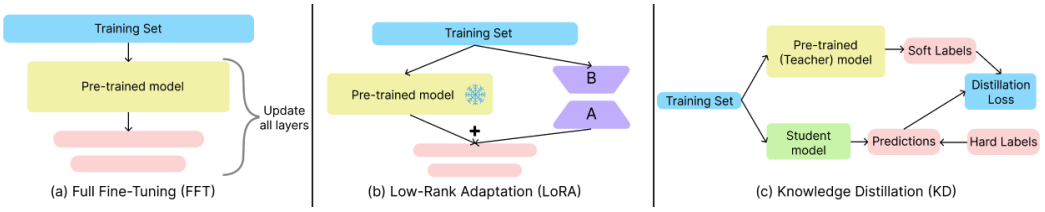

Figure 1: Overview of three fine-tuning methods: (a) FFT, which updates all model parameters; (b) LoRA, which adds low-rank matrices to update a small subset of parameters; and (c) KD, which trains a smaller student model to emulate a larger teacher model.

2 Method

We propose KD-LoRA, a novel fine-tuning methodology that integrates LoRA with KD. The proposed method involves three main steps: (1) selecting and fine-tuning a teacher model, (2) initializing a smaller student model with LoRA modules, and (3) performing distillation to transfer knowledge from the teacher model to the student model.

Teacher Model. Let $\mathcal{T}$ denote the teacher model, initialized from a pre-trained language model (e.g., BERT, RoBERTa, DeBERTa). The teacher model is fine-tuned on a specific task $\mathcal{D}_{\text{task}}$, using FFT, where all parameters of the model are updated (Lv et al., 2024). The objective function for fine-tuning the teacher model is:

$$\mathcal{L}_{\text{task}}^{\mathcal{T}} = \frac{1}{|\mathcal{D}_{\text{task}}|} \sum_{(x_i, y_i) \in \mathcal{D}_{\text{task}}} \mathcal{L}_{\text{CE}}(\mathcal{T}(x_i), y_i) \tag{1}$$

where $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss (CEL) loss, x_i represents the input data, and y_i denotes the corresponding label. This loss function measures the discrepancy between the predicted probabilities $\mathcal{T}(x_i)$ and the true labels y_i. The fine-tuned teacher model $\mathcal{T}$ then serves as the source of distilled knowledge.

Student Model with LoRA. The student model $\mathcal{S}$ is initialized from a smaller version within the same model family as the teacher model $\mathcal{T}$. We modify the student model by injecting LoRA modules into its architecture. Specifically, LoRA is applied to the attention layers, where the weight matrices W_q and W_v (corresponding to the query and value projections) are decomposed as follows:

$$W_q = W_q^{\text{base}} + A_q B_q, \quad W_v = W_v^{\text{base}} + A_v B_v \tag{2}$$

where W_q^{base} and W_v^{base} are the pre-trained weight matrices, while A_q, B_q, A_v, and B_v are the low-rank matrices, the only parameters updated during fine-tuning.

KD-LoRA. With LoRA modules already in place, the KD process is performed, where the student model $\mathcal{S}$ learns from the teacher model. During this phase, the student model, equipped with LoRA, adapts its low-rank matrices to capture the knowledge transferred from the teacher. The student model is trained on the target task $\mathcal{D}_{\text{task}}$ using the combined loss function $\mathcal{L}_{\text{total}}^{\mathcal{S}}$, which is given by:

$$\mathcal{L}_{\text{total}}^{\mathcal{S}} = \alpha \mathcal{L}_{\text{task}}^{\mathcal{S}} + (1 - \alpha) \mathcal{L}_{\text{KD}}(z^{\mathcal{S}}, z^{\mathcal{T}}) \tag{3}$$

where $z^{\mathcal{T}}$ and $z^{\mathcal{S}}$ are the logits (outputs before the softmax layer) of the teacher and student models, respectively. The KD loss $\mathcal{L}_{\text{KD}}$ is computed as the Kullback-Leibler divergence (KL) (Shlens, 2014) between the softened output probabilities of the teacher model $\mathcal{T}$ and the student model $\mathcal{S}$. The parameter α balances the task-specific loss $\mathcal{L}_{\text{task}}^{\mathcal{S}}$ and the KD loss $\mathcal{L}_{\text{KD}}$. During each training step, the student model's low-rank matrices are updated to minimize the loss in Eq. 3.

3 Experiments

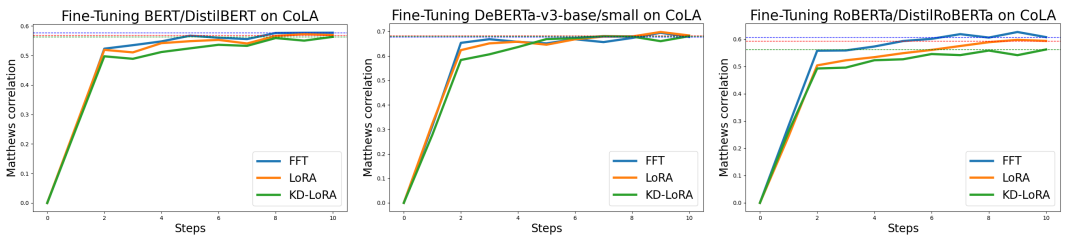

Figure 2: Comparison of convergence speed between full fine-tuning (FFT), LoRA, and KD-LoRA for three LLMs on the CoLA task. **KD-LoRA matches the convergence speed of FFT and LoRA.**

Table 1: Performance metrics for BERT-base (BERT-b), DeBERTa-v3-base (DeB-b), and RoBERTa-base (RoB-b) across GLUE tasks using three fine-tuning methods. Results show the median of the top 6 hyperparameter and PEFT setups. DistilBERT-base (DBERT-b), DeBERTa-v3-small (DeB-s), and DistilRoBERTa-base (DRoB-b) serve as student models. Metrics include Matthews correlation for CoLA, average Pearson/Spearman correlations for STS-B, average accuracy/F1 scores for MRPC and QQP, and accuracy for all other tasks. **KD-LoRA achieves about 97% of FFT's performance and about 98% of LoRA's performance.**

Model	**BERT-b**		**DBERT-b**	**DeB-b**		**DeB-s**	**RoB-b**		**DRoB-b**
Method	FFT	LoRA	KD-LoRA	FFT	LoRA	KD-LoRA	FFT	LoRA	KD-LoRA
CoLA	57.7	56.9	56.3	67.8	69.1	68.1	60.9	59.4	56.8
MNLI_m	84.5	83.4	82.0	90.3	90.3	88.8	87.7	87.2	83.3
MNLI_{mm}	84.9	83.9	82.4	90.6	90.2	89.0	87.4	86.9	83.4
MRPC	89.0	89.2	88.3	91.9	90.9	90.7	91.1	89.9	89.3
QNLI	91.8	91.1	89.7	94.1	94.3	93.4	92.7	92.8	90.7
QQP	89.7	87.9	89.1	91.2	90.4	89.9	89.8	88.6	87.3
RTE	71.6	70.1	64.0	85.0	84.0	78.8	74.8	71.8	65.3
SST-2	92.8	92.6	92.0	95.9	96.0	95.7	94.3	94.2	92.9
STS-B	89.5	88.9	88.7	91.5	91.1	89.8	90.8	90.3	87.9
WNLI	56.3	56.9	56.3	66.9	56.3	56.3	56.3	56.3	56.3
Score	**80.8**	**80.1**	**78.9**	**86.5**	**85.3**	**84.1**	**82.6**	**81.7**	**79.3**

Table 2: Comparison of trainable parameters, memory usage, and inference time for three fine-tuning methods across three models and their distilled counterparts for KD-LoRA. Inference time is averaged over 100 runs on the CoLA validation set. **With a rank of 8, KD-LoRA reduces trainable parameters by 99% compared to FFT and 49% compared to LoRA, while lowering GPU memory usage by 75% and 30%, respectively. KD-LoRA also cuts inference time by 30%.**

Model	**Method**	**Rank 8**	**Rank 16**	**Rank 32**	**Rank 64**	**Memory Usage**	**Inference Time**
BERT-base	FFT	110M	110M	110M	110M	1332.0MB	6.10s
	LoRA	2.9M	5.9M	11.8M	23.6M	463.5MB	6.22s
DistilBERT-base	KD-LoRA	1.2M	2.4M	4.7M	9.4M	**296.8MB**	**5.36s**
RoBERTa-base	FFT	125M	125M	125M	125M	1515.9MB	7.21s
	LoRA	2.9M	5.9M	11.8M	23.6M	531.9MB	7.19s
DistilRoBERTa-base	KD-LoRA	1.5M	2.9M	5.9M	11.8M	**358.3MB**	**4.44s**
DeBERTa-v3-base	FFT	183M	183M	183M	183M	2234.5MB	14.37s
	LoRA	2.9M	5.9M	11.8M	23.6M	763.4MB	15.62s
DeBERTa-v3-small	KD-LoRA	1.5M	2.9M	5.9M	11.8M	**590.3MB**	**10.38s**

For our experiments, we select three widely recognized encoder-only LLMs: BERT (Devlin et al., 2019), RoBERTa (Zhuang et al., 2021), and DeBERTaV3 (He et al., 2023). We evaluate three fine-tuning strategies across these models on the GLUE benchmark: FFT, LoRA, and KD-LoRA. In this approach, we employ compact student models that belong to the same family as their corresponding larger teacher models. Specifically, we use DistilBERT-base (Sanh et al., 2020), DeBERTa-v3-small, and DistilRoBERTa-base as student models for BERT-base, DeBERTa-v3-base, and RoBERTa-base, respectively. For FFT, we select 25 hyperparameter configurations, varying learning rates (2e-5 to 5e-5), batch sizes (8 to 32), epochs (2 to 5), and weight decay (0.01 to 0.1). For LoRA and KD-LoRA, we select 24 PEFT configurations, varying rank (8 to 32), epochs (3 to 5), LoRA alpha (16 to 32), and LoRA dropout (0.0 to 0.1). All experiments are conducted using NVIDIA A100 GPUs. Table 1 shows the results calculated based on the median of the top 6 configurations. Table 2 provides the number of trainable parameters for each method at different ranks, along with their GPU memory usage during inference and the inference time calculated on the CoLA dataset.

KD-LoRA achieves approximately 97% of FFT's performance and about 98% of LoRA's, with scores of 78.9 for the student model of BERT-base compared to 80.8 for FFT and 80.1 for LoRA. It reduces the number of trainable parameters by about 99% compared to FFT and about 49% compared to LoRA, updating 1.5M parameters in the DistilRoBERTa-base model with KD-LoRA versus 2.9M with LoRA at a rank of 8. KD-LoRA also reduces GPU memory usage by 75% compared to FFT and 30% compared to LoRA, resulting in a model that is about 40% more compact than both FFT and

LoRA. Additionally, KD-LoRA decreases inference time by around 30% on the CoLA dataset, while maintaining comparable convergence speed, as illustrated in Figure 2.

4 Conclusion

We present KD-LoRA, a novel fine-tuning method that integrates LoRA modules into a student model while leveraging KD from a larger teacher model. Empirical results on the GLUE benchmark show that KD-LoRA retains approximately 97% of FFT performance and 98% of LoRA performance, all while reducing model size by around 40%. KD-LoRA also lowers trainable parameters by 99% compared to FFT and 49% compared to LoRA, reduces GPU memory usage by 75% compared to FFT and 30% compared to LoRA, and cuts inference time by 30%.

Acknowledgements

The authors thank CMLabs, Mila and CIFAR for research funding. This research was enabled by the compute provided by Calcul Quebec and Digital Research Alliance of Canada.

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, aug 2020. ISSN 2150-8097. doi: 10.14778/3415478.3415530. URL https://doi.org/10.14778/3415478.3415530.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476209. URL https://doi.org/10.1145/3458817.3476209.

Sajal Dash, Isaac Lyngaas, Junqi Yin, Xiao Wang, Romain Egele, Guojing Cong, Feiyi Wang, and Prasanna Balaprakash. Optimizing distributed training on frontier for large language models. *CoRR*, abs/2312.12705, 2023. doi: 10.48550/ARXIV.2312.12705. URL https://doi.org/10.48550/arXiv.2312.12705.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized BERT pre-training approach with post-training. In Sheng Li, Maosong Sun, Yang Liu, Hua Wu, Kang Liu, Wanxiang Che, Shizhu He, and Gaoqi Rao, editors, *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China, August 2021. Chinese Information Processing Society of China. URL https://aclanthology.org/2021.ccl-1.108.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9, 2023. doi: 10.1109/HPEC58863.2023.10363447.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning, 2023. URL https://arxiv.org/abs/2303.15647.

Arjun Singh, Nikhil Pandey, Anup Shirgaonkar, Pavan Manoj, and Vijay Aski. A study of optimizations for fine-tuning large language models, 2024. URL https://arxiv.org/abs/2406.02290.

Taeho Kim, Yanming Wang, Vatshank Chaturvedi, Lokesh Gupta, Seyeon Kim, Yongin Kwon, and Sangtae Ha. Llmem: Estimating gpu memory usage for fine-tuning pre-trained llms. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 6324–6332. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/699. URL https://doi.org/10.24963/ijcai.2024/699. Main Track.

Yanjie Dong, Xiaoyi Fan, Fangxin Wang, Chengming Li, Victor C. M. Leung, and Xiping Hu. Fine-tuning and deploying large language models over edges: Issues and approaches, 2024. URL https://arxiv.org/abs/2408.10691.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=lq62uWRJjiY.

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, 2024. URL https://openreview.net/forum?id=FAO4VS9QRV.

Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. Analyzing and reducing catastrophic forgetting in parameter efficient tuning, 2024. URL https://arxiv.org/abs/2402.18865.

Hongbo Zhao, Bolin Ni, Junsong Fan, Yuxi Wang, Yuntao Chen, Gaofeng Meng, and Zhaoxiang Zhang. Continual forgetting for pre-trained vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28631–28642, June 2024a.

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. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32100–32121. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bn.html.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

George Pu, Anirudh Jain, Jihan Yin, and Russell Kaplan. Empirical analysis of the strengths and weaknesses of PEFT techniques for LLMs. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023. URL https://openreview.net/forum?id=HB7zDQ4mvX.

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=APuPRxjHvZ.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. URL https://arxiv.org/abs/1604.06174.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning, 2023b. URL https://arxiv.org/abs/2308.03303.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *5th Workshop on practical ML for limited/low resource settings*, 2024b. URL https://openreview.net/forum?id=AzqPyO22zt.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986.

Baohao Liao, Yan Meng, and Christof Monz. Parameter-efficient fine-tuning without introducing new latency. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4242–4260, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.233. URL https://aclanthology.org/2023.acl-long.233.

Jihao Gu, Shuai Chen, Zelin Wang, Yibo Zhang, and Ping Gong. Sara: Singular-value based adaptive low-rank adaption, 2024a. URL https://arxiv.org/abs/2408.03290.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, et al. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2402.13116.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372. URL https://aclanthology.org/2020.findings-emnlp.372.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8187–8198, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.445.

Jonathon Shlens. Notes on kullback-leibler divergence and likelihood, 2014. URL https://arxiv.org/abs/1404.2000.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.