

DoRA the Explorer

Stanford CS224N Default Project

Cayden Gu

Department of Computer Science
Stanford University
caydengu@stanford.edu

Imogen Lee

Department of Computer Science
Stanford University
imlee@stanford.edu

Abstract

Fine-tuning large language models for downstream tasks is becoming an increasingly challenging task due to computational cost and memory requirements associated with updating billions of parameters. Weight-based Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA and DoRA address this challenge by restricting updates to low-rank adaptations of pretrained weights. Representation-space approaches like LoReFT offer an alternative strategy to this challenge by intervening on hidden representations. In this study, we implement and compare these methods on GPT-2 across three diverse tasks: paraphrase detection, sonnet generation, and sentiment analysis. We validate the effectiveness and efficiency of PEFT approaches as reported in prior work, including the ability of LoRA to match full fine-tuning performance under appropriate hyperparameter configurations and the improved accuracy of DoRA through the decomposition of magnitude and directional weight updates. LoReFT achieves high efficiency but is more sensitive to intervention choices and model scale. These findings highlight trade-offs between parameter efficiency, robustness, and scalability among PEFT approaches for natural language processing (NLP) tasks.

1 Key Information to Include

- Mentor: Simon Kim
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

In natural language processing, standard transfer learning traditionally involves pre-training a model on general domain data followed by adaptation to various downstream tasks. This adaptation process is usually carried out through full fine-tuning; however, this approach has become increasingly infeasible as model sizes have exploded. A primary example of this is GPT-3, a model that features 175 billion parameters [1]. For large models, higher computational costs and the critical deployment challenge of requiring each specific task to have a unique copy of the original model have necessitated a shift towards Parameter Efficient Fine-Tuning (PEFT) methods.

The prohibitive costs associated with full fine-tuning have made Low-Rank Adaptation (LoRA) an attractive alternative. Hu et al. (2021) showed that freezing pre-trained weights and injecting trainable rank decomposition matrices allows for significant parameter reduction coupled with zero inference latency [2]. Weight-Decomposed Low-Rank Adaptation (DoRA) builds upon LoRA by investigating the update patterns of full fine-tuning, as Liu et al. (2024) found that full fine-tuning exhibits a negative correlation between magnitude and directional updates, whereas LoRA is rigidly positive [3]. By decoupling weights into magnitude (m) and direction (V) components, DoRA applies low-rank updates specifically to direction, enhancing learning capacity and stability [3].

Moreover, prior interpretability work has suggested that representations encode rich semantic information, presenting a powerful alternative to overcome the limitations of weight-based adaptation methods (Wu et al., 2024) [4]. Unlike the weight modifications of LoRA and DoRA, Low-rank Linear Subspace Representation Fine-Tuning (LoReFT) learns task-specific interventions on hidden representations during the model’s forward pass. LoReFT is 15x–65x more parameter-efficient than LoRA [4], serving as a comparative goal for us to assess whether representation fine-tuning methods may be more effective for tasks like sentiment analysis, paraphrase detection, and sonnet generation.

In this study, we implement and compare three PEFT methods (namely, LoRA, DoRA, and LoReFT) on GPT-2 [5] across paraphrase detection, sonnet generation, and sentiment analysis to evaluate their ability to match the performance of full fine-tuning while training far fewer parameters. Our experiments show that weight-based PEFT methods such as LoRA and DoRA can closely match full fine-tuning performance while training only 1–2% of model parameters, with DoRA achieving comparable results to LoRA at roughly half the rank due to its magnitude-direction decomposition. In contrast, the representation-based method LoReFT achieves high efficiency ($\sim 0.06\%$ parameters) but is substantially more sensitive to intervention design and scale, showing large performance variation across token-position configurations (27.38 vs. 41.79 chrF on sonnet generation) and a notable dev-test generalization gap when scaled up to GPT-2 Large.

3 Related Work

Adapter-based methods. As one solution to the higher computational costs of FullIFT, adapter layers introduced lightweight, trainable modules between the layers of pretrained language models (LM) [6]. While these methods demonstrated strong performance with minimal parameter growth, adapter modules also introduced extra sequential computations that led to increased inference latency [6].

LoRA and DoRA. These two methods represent the current dominant approach to PEFT and function by using low-rank matrices to approximate weight updates during training [4]. Recent work in Schulman & Thinking Machines Lab (2025) finds that for datasets that exceed LoRA capacity, LoRA underperforms full fine-tuning [7]. In this light, Schulman & Thinking Machines Lab (2025) have established a “low-regret regime,” where LoRA matches full fine-tuning efficiency provided it is applied to all layers (particularly, MLP models) instead of only attention layers. [7].

Representation editing. In response to limitations in prompt engineering and fine-tuning approaches, activation engineering has emerged as a method for controlling language model behavior by directly modifying internal representations. One common technique involves computing a steering vector from intermediate activations of the model [8]. These vectors can then be computed from intermediate activations and subsequently added to the hidden states of the language model during inference to steer the model’s outputs toward desired outcomes [8].

4 Approach

The primary goal of our paper is to determine whether PEFT methods can replicate the performance of full-parameter fine-tuning while significantly reducing the number of trainable parameters. We focus our investigation on two types of adaptations: weight-update adaptations (e.g., LoRA and DoRA) and representation-intervention adaptations (e.g., LoReFT). Figure 4.2 provides a schematic overview of our three extensions.

4.1 Baseline GPT-2 model

To start, we implement the standard GPT-2 model by following the three-step approach outlined in the project handout. We first instantiate the multi-head self-attention and then construct the full GPT transformer layers in two blocks (an attention block and a feedforward block). We also implemented the token and positional embedding functions within the model class. We then wrote the step function for the Adam optimizer [9], which tracks m_t , the exponential moving average of the gradients, and v_t , the exponential moving average of the squared gradients to capture their magnitude. Bias correction is applied using the efficient variant as described in the default project handout, and decoupled weight decay is incorporated to improve regularization [10].

4.2 PEFT methods

LoRA. For LoRA, we freeze the pre-trained GPT-2 weights and inject trainable low-rank matrices into the target attention modules. The weight update ΔW is decomposed as $W_0 + \Delta W = W_0 + BA$, where $W_0 \in \mathbb{R}^{d \times k}$ functions as the frozen pre-trained weight matrix. $A \in \mathbb{R}^{r \times k}$ is initialized with Kaiming Uniform initialization and $B \in \mathbb{R}^{r \times r}$ is initialized to zero, as described in Hu et. al (2021) [2]. Both A and B contain trainable parameters.

DoRA. For DoRA, we modify the forward pass of the linear layers to separate magnitude and direction by initializing $m = \|W_0\|_c$ (column-wise norm) and freeze W_0 . The directional component is updated using standard LoRA matrices B and A , while m is trained as a separate parameter vector. The adapted weight W' is then computed as $W' = m \odot \frac{W_0 + BA}{\|W_0 + BA\|_c}$, as described in Liu et al. (2024) [3].

LoReFT. In contrast to the weight-modification approaches of LoRA and DoRA, we next explored LoReFT, a parametrization of ReFT that intervenes on hidden representations in the linear subspace [4]. LoReFT is defined by eq. (1):

$$\Phi_{\text{LoReFT}}(\mathbf{h}) = \mathbf{h} + \mathbf{R}^T(\mathbf{W}\mathbf{h} + \mathbf{b} - \mathbf{R}\mathbf{h}) \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{r \times d}$ represents a linear projection and $\mathbf{b} \in \mathbb{R}^r$ represents a bias vector. For LoReFT, we freeze the parameters of the LM and the learned parameters consist of $\phi = \{\mathbf{R}, \mathbf{W}, \mathbf{b}\}$.

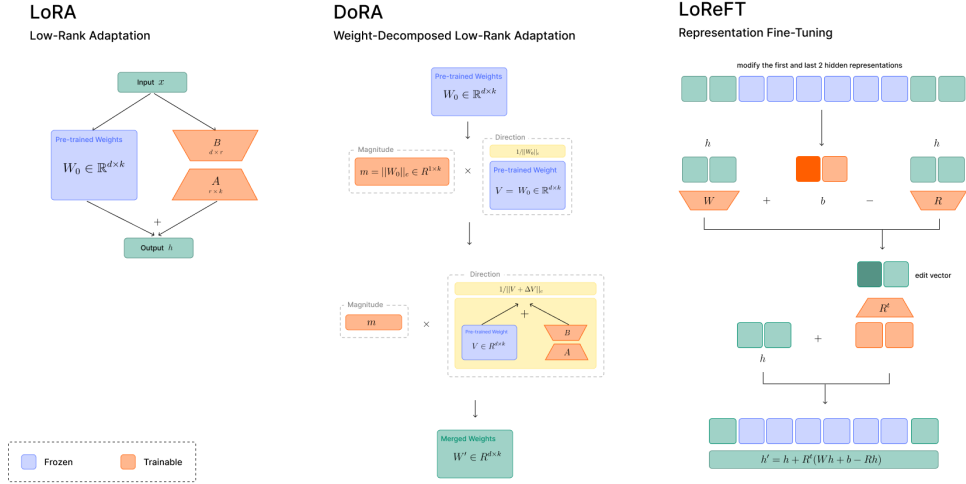


Figure 1: **Illustration of three parameter-efficient adaptation methods.** 1) LoRA applies low-rank updates to model weight matrices, 2) DoRA decomposes weight updates into magnitude and directional components, and 3) LoReFT intervenes directly on hidden representations by modifying activations within a learned low-rank subspace.

4.3 Baselines

As baselines, we rely on the following:

1. **Zero-Shot GPT-2 Small (Lower Bound):** We evaluate the pre-trained GPT-2 model to establish a floor for performance without fine-tuning.
2. **Full Fine-Tuning (Upper Bound):** As implemented in the default project, we update all parameters of GPT-2. This serves as the performance ceiling, allowing us to evaluate how close our PEFT techniques can reach.

3. **Standard LoRA:** We first implement LoRA that only adapts the attention weights, as described in Hu et al. (2021) [2], which serves as our primary competitive baseline for PEFT methods (including LoRA on all linear layers). We denote this baseline LoRA as LoRA[†].

4.4 Implementation

We build on the CS224N starter code for our GPT-2 backbone. For LoRA and DoRA, although we indicated in the Milestone that we planned to use our custom implementation of LoRA, we instead used the HuggingFace PEFT library [11] for all reported experiments due to its efficiency and modularity, allowing us to focus our contributions on the comparative analysis across methods, tasks, and model sizes. For LoReFT, we implement the method from scratch based on Wu et al. (2024) [4].

5 Experiments

5.1 Evaluation method

We evaluate LoRA, DoRA, and LoReFT on the three downstream tasks from the default final project: Cloze-Style Paraphrase Detection, Sonnet Generation, and Sentiment Analysis. We rely on the standard evaluation metrics, namely, accuracy for paraphrase detection and sentiment analysis and chrF score for sonnet generation.

5.2 Data

We utilize the datasets provided in the default final project without additional data collection. As summarized in Table 1, our benchmarks include 400,000 Quora Question Pairs for binary classification; 154 Shakespearean Sonnets with 14 lines each; the Stanford Sentiment Treebank (SST) dataset, which consists of 11,855 single sentences rated on a five-point scale; and CFIMDB, which features 2,434 highly polar movie reviews classified into binary negative or positive categories.

Table 1: Overview of datasets used in this study

Dataset	Task	Format	No. of Examples
Quora Question Pairs	Paraphrase Detection	Question Pairs	400,000
Shakespearean Sonnets	Sonnet Generation	Sonnets	154
Stanford Sentiment Treebank	Sentiment Analysis	Sentences	11,855
CFIMDB	Sentiment Analysis	Movie Reviews	2,434

5.3 Experimental details

We ran experiments on GPT-2 Small, Medium, and Large for all the PEFT methods described in Section 4.2 (namely LoRA, DoRA, and LoReFT), along with the baselines from Section 4.3. Full experimental results can be found in Appendix B (paraphrase detection), Appendix D (sonnet generation), and Appendix E (sentiment analysis).

We focused our hyperparameter tuning primarily on paraphrase detection using GPT-2 Small. We began by running baseline experiments: zero-shot GPT-2 (0.390 dev accuracy), full fine-tuning with a brief learning rate sweep (best: $2e-4$ with 0.897 dev accuracy), and LoRA[†] applied only on attention layers (rank 8, $lr=1e-5$, 0.849 dev accuracy). The gap between LoRA[†] and full-finetuning motivated us to target all linear layers (attention and MLP layers), following the work by Schulman & Thinking Machines Lab (2025) [7].

We then tuned the hyperparameters for each PEFT method, as detailed in Appendix C. For both LoRA and DoRA, we followed the recommendations from Schulman & Thinking Machines Lab (2025) [7]. Inspired by Liu et al. (2024) [3], who show that DoRA at half the rank of LoRA can match or exceed LoRA’s accuracy, we then followed half-rank configurations relative to LoRA for DoRA. For LoReFT, we referred to the hyperparameter recommendations in Wu et al. (2024) [4] Appendix D.2, starting with rank 4 (higher rank does not always bring performance gain [4]) and then sweeping learning rates over the same grid. We then performed ablation studies on layer selection (all 12

Table 2: **Summary of best results across all tasks and model sizes.** Each method uses its best hyperparameters (Table 3) from sweeps on GPT2-Small for paraphrase detection (Appendix C). LoRA[†] denotes attention-only LoRA (baseline); LoRA[★] and DoRA[★] target all linear layers. LoReFT[★] intervenes on all tokens for the sonnet generation task; intervenes on f4+l4 token positions otherwise. Full experiment details are provided in Appendices B, D, E.

Method	Rank	Params (%)	Quora Dev Acc	Sonnet Dev chrF	SST Dev Acc	CFIMDB Dev Acc
<i>Baselines on GPT-2 Small</i>						
Zero-shot	—	0%	0.390	40.25	0.195	0.502
Full FT	—	100%	0.897	42.33	0.510	0.976
LoRA [†]	8	~0.35%	0.675	41.46	0.293	0.592
<i>PEFT Methods on GPT-2 Small</i>						
LoRA [★]	16	2.08%	0.893	41.79	0.507	0.976
DoRA [★]	8	1.12%	0.893	41.96	0.506	0.980
LoReFT [★]	4	~0.06%	0.853	41.79	0.520	0.976
<i>Full FT + PEFT Methods on GPT-2 Medium</i>						
Full FT	—	100%	0.906	42.55	0.544	0.971
LoRA [★]	16	1.95%	0.903	42.63	0.556	0.980
DoRA [★]	8	1.05%	0.902	42.23	0.558	0.967
LoReFT [★]	4	~0.06%	0.883	42.49	0.543	0.967
<i>Full FT + PEFT Methods on GPT-2 Large</i>						
Full FT	—	100%	0.910	42.93	0.573	0.976
LoRA [★]	16	1.68%	0.903	42.92	0.550	0.971
DoRA [★]	8	0.90%	0.905	42.91	0.550	0.984
LoReFT [★]	4	~0.05%	0.888	42.28	0.546	0.963

layers vs layers 0-5 vs layers 6-11) along with which positions to intervene on (last-1, first-1+last-1, first2+last-2, first-4+last-4), finding that all layer interventions at first-4+last-4 positions performed best (0.853 dev accuracy at only 0.06% of parameters) on paraphrase detection.

Our final hyperparameters can be found in Table 3 in Appendix B. We denote our tuned versions of LoRA, DoRA, and LoReFT as LoRA[★], DoRA[★], and LoReFT[★], respectively.

Finally, to reduce computational costs, we transferred the tuned hyperparameters to GPT-2 Medium and Large without additional tuning. This transfer is motivated (in part) by Schulman & Thinking Machines Lab (2025) [7], who find that the optimal LoRA learning rate is approximately rank-independent and transfers consistently across model sizes. While the optimal rank could vary with model size, our results below remained competitive with full fine-tuning across both larger models; we additionally transferred the same hyperparameters to the sonnet generation and sentiment analysis tasks, where we again noticed that the hyperparameters transferred reasonably well (see Section 5.4).

5.4 Results

As a default project team, we made three submissions to the test leaderboard for both paraphrase detection and sonnet generation for our fully fine-tuned model, DoRA[★] model, and LoReFT[★] model (all on GPT-2 Large). Fully fine-tuned scored an accuracy of **0.908** on accuracy for paraphrase detection and a chrF score of **42.360** for sonnet generation. DoRA[★] scored an accuracy of **0.886** on accuracy for paraphrase detection and a chrF score of **41.654** for sonnet generation. Finally, LoReFT[★] scored an accuracy of **0.787** on accuracy for paraphrase detection and a chrF score of **41.833** for sonnet generation.

We also made numerous submissions to the dev leaderboard, achieving a maximum Paraphrase Detection dev accuracy of **0.910** and Sonnet Generate dev chrF of **42.93** after additional fine-tuning.

Overall, we saw that the PEFT methods of LoRA, DoRA, and LoReFT significantly improved performance over baselines, especially zero-shot GPT-2, with very high parameter efficiency (see Table 2). Each method was able to achieve a similar level of performance as fully fine-tuned models across all three tasks on the development sets on GPT-2 Small and Medium, demonstrating that the hyperparameters we tuned for GPT-2 small were able to transfer to new model sizes and tasks

reasonably well. However, as model size increases to GPT-2 Large, the performance gap between PEFT methods and full fine-tuning widens (though still far surpasses baselines), suggesting that considerably larger model sizes require different hyperparameters for PEFT methods.

We observed that DoRA $\star$ matched or exceeded LoRA $\star$ across nearly all tasks despite using roughly half the rank (8 vs. 16) and fewer parameters (1.12% vs. 2.08% on GPT-2 Small; 1.05% vs. 1.95% on GPT-2 Medium), consistent with Liu et al.’s (2024) finding that magnitude-direction decomposition does improve parameter efficiency.

Although we observed that these PEFT techniques were able to achieve the same level of performance as fully fine-tuned models across all three tasks on dev sets, we were surprised by the test accuracy for LoReFT $\star$ on paraphrase detection: its dev accuracy on GPT-2 Large (0.888) was within 0.022 of full fine-tuning (0.910), but its test accuracy dropped to 0.787, showing a gap of 0.101 from dev accuracy to test accuracy. By comparison, DoRA $\star$ only dropped by 0.019 from dev accuracy to test accuracy, and full fine-tuning only had a gap of 0.002. We examine potential reasons for this result in Section 6.3.

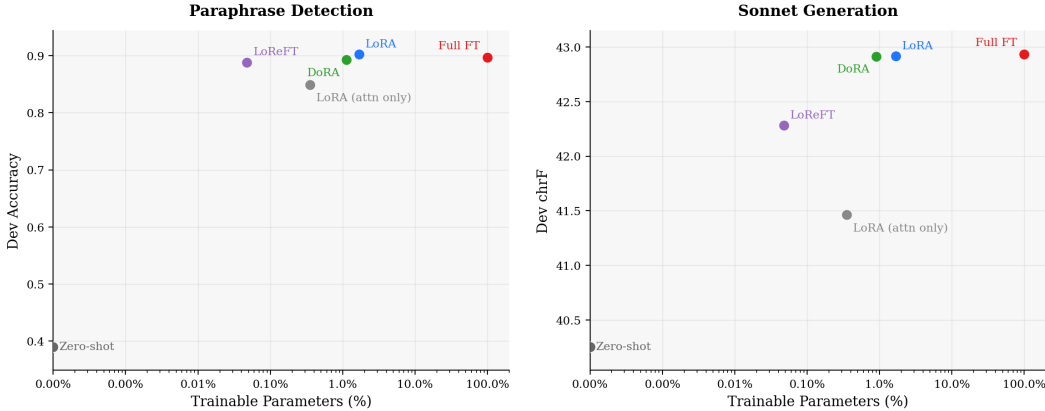
6 Analysis

6.1 Properly tuned PEFT methods match full fine-tuning performance

Schulman & Thinking Machines Lab [7] demonstrate that LoRA can match full fine-tuning performance when hyperparameters are properly tuned. We observe a similar trend across all three PEFT methods on GPT-2 Small (Figure 2). On paraphrase detection, LoRA $\star$ and DoRA $\star$ approach full fine-tuning accuracy (0.893 and 0.893 vs. 0.897) at 2% and 1% of parameters respectively, while LoReFT $\star$ reaches 0.853 at just 0.06%. On sonnet generation, all tuned methods cluster within 0.6 chrF of full fine-tuning. Finally, LoRA $\star$ matches full fine-tuning on SST (0.507 vs. 0.510) and CFIMDB (0.976 vs. 0.976), while DoRA $\star$ achieves nearly identical results at half the rank. Notably, attention-only LoRA (LoRA $\dagger$) fails to match LoRA $\star$ on all linear layers despite similar parameter counts, validating the recommendation to target all linear layers [7].

This pattern holds across model scales. Even on medium and large models, LoRA $\star$ and DoRA $\star$ remain close in performance to full fine-tuning across most tasks, demonstrating that careful hyperparameter tuning allows PEFT to closely approximate full model performance at a fraction of the computational cost. As Figure 2 demonstrates, all three tuned PEFT methods substantially outperform the zero-shot and LoRA $\dagger$ while approaching full fine-tuning’s performance with significantly fewer parameters.

Figure 2: Parameter efficiency vs. dev performance on GPT-2 Small for paraphrase detection and sonnet generation. Each point represents a method’s best dev performance plotted against its trainable parameter percentage (log-scaled). Attention-only LoRA (LoRA $\dagger$) underperforms LoRA applied to all linear layers. Similar plots for SST and CFIMDB are in Appendix A.

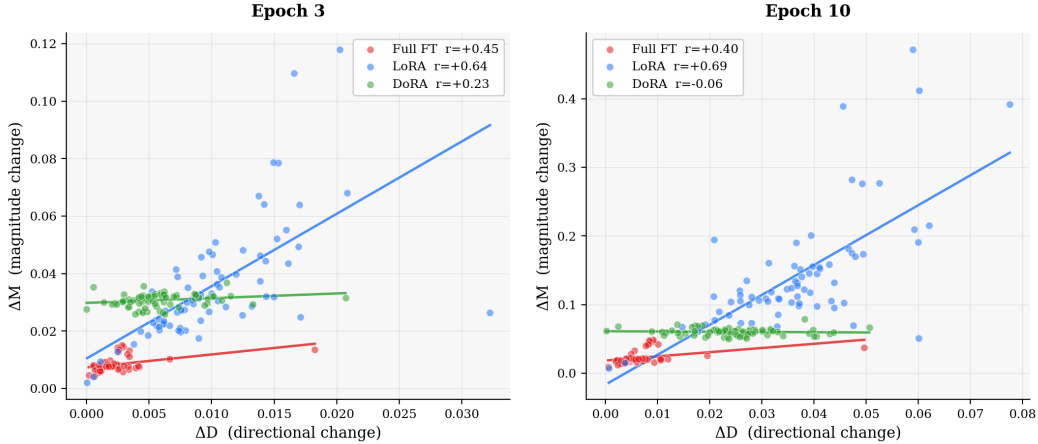


6.2 DoRA decouples magnitude and directional weight updates

To understand how each method adapts pretrained weights, we reproduce the weight decomposition analysis of Liu et al. (2024) [3] on GPT-2 Small for the paraphrase detection task. To track how update patterns evolve during training, we checkpoint at epoch 3 in addition to the final epoch 10. For that checkpointed model and the final model at epoch 10, we decompose all 72 weight matrices (6 linear layers $\times$ 12 transformer blocks) into per-column magnitude and direction components and measure each matrix’s mean magnitude change (ΔM) and mean directional change (ΔD) relative to the pretrained weights (Figure 3).

LoRA exhibits a strong positive ΔM - ΔD correlation throughout training ($r = +0.69$ at epoch 10), indicating that magnitude and direction updates are tightly coupled (i.e. when LoRA makes a large directional change, it also makes a proportionally large magnitude change). On the other hand, we observe that DoRA decouples these two components. Its correlation drops from $r = +0.23$ at epoch 3 to $r = -0.06$ by epoch 10. This validates that DoRA’s explicit magnitude-direction decomposition allows each component to be tuned independently, enabling smaller, more targeted adjustments than LoRA’s coupled updates. This property helps explain DoRA’s competitive performance at roughly half the parameter count of LoRA (Tables 5, 9). Full fine-tuning shows a moderate positive correlation ($r = +0.40$), which diverges from the negative correlation reported by Liu et al. (2024) [3], which we attribute to the difference in model and task.

Figure 3: **DoRA decouples magnitude and direction updates.** Each point represents one of 72 weight matrices, plotted by its mean magnitude change (ΔM) vs. mean directional change (ΔD) relative to pretrained weights, and we draw the line of best-fit for each model type. By epoch 10, LoRA’s strong positive slope contrasts with DoRA’s near-flat relationship.



6.3 Layer and Position Selection Matter for LoReFT and are Task and Scale Dependent

Based on Liu et al. (2024)’s suggestions on choosing hyperparameters for ReFT, we analyze the sensitivity of ReFT to hyperparameter configurations [3]. As shown in Table 8, the optimal configuration for paraphrase detection on GPT-2 Small was to intervene on all layers at multiple token positions. The layer ablation indicates that the top six layers (6–11) outperform the bottom six layers (0–5) on the paraphrase task (0.838 vs. 0.834), consistent with our expectation that higher layers encode more relevant information for classification tasks. The position ablation shows that f2+l2 and f4+l4 perform nearly identically (0.8532 vs. **0.8533**), suggesting diminishing returns from incorporating additional token positions beyond the first and last few tokens in classification tasks.

However, on sonnet generation, we noticed that restricting interventions to f4+l4 caused a sharp drop in chrF compared to intervening on all positions (27.38 vs. **41.79**). In generation tasks, since each generated token becomes the input for the next step, interventions that only modify boundary positions during training may cause the learned adaptation signal to decay with each generation step.

Additionally, as discussed in Section 5.4, LoReFT $\star$ on GPT-2 Large showed a 0.101 drop from dev to test accuracy on paraphrase detection. We hypothesize that LoReFT’s first-4 and last-4 fixed-position intervention did not transfer well from GPT-2 Small (12 layers; 768 hidden dimension) to GPT-2 Large (36 layers; 1280 hidden dimension), suggesting that the optimal number of intervention positions may need to scale with model depth. Second, we did not apply weight decay or dropout to LoReFT’s intervention parameters, which we believe may have allowed overfitting. Future work should investigate both intervening on different token positions when scaling model size and regularization strategies for representation-based fine-tuning.

7 Conclusion

Our experiments demonstrate that weight-based PEFT methods, such as LoRA and DoRA, can closely match full fine-tuning performance while training only 1-2% of the parameters, with DoRA achieving comparable results at half the rank of LoRA thanks to its magnitude-direction decomposition. LoReFT achieved remarkable efficiency at $\sim 0.06\%$ of parameters but faced sensitivity to which layers and token positions are intervened on, especially on different tasks (27.38 vs 41.79 chrF gap on sonnet generation with f4+l4 vs all positions) and model sizes (0.101 drop from dev to test accuracy on paraphrase detection when scaling to GPT-2 Large).

The primary limitation we faced was limited compute, which led us to only hyperparameter tune on GPT-2 Small for paraphrase detection and extrapolate the results to larger models and other tasks. Moreover, as discussed in Section 6.3, with more resources we would evaluate intervening on different token positions and layers for different tasks and model sizes, similar to our ablation studies in Table 8 for LoReFT on GPT-2 small. Finally, future work could investigate combining both weight-based and representation-based PEFT adaptations, since they fundamentally modify different parts of GPT-2.

8 Team Contributions

Cayden: Implemented the base GPT-2 model, conducted research, trained models and managed experimentation, and wrote the experiments section and appendices.

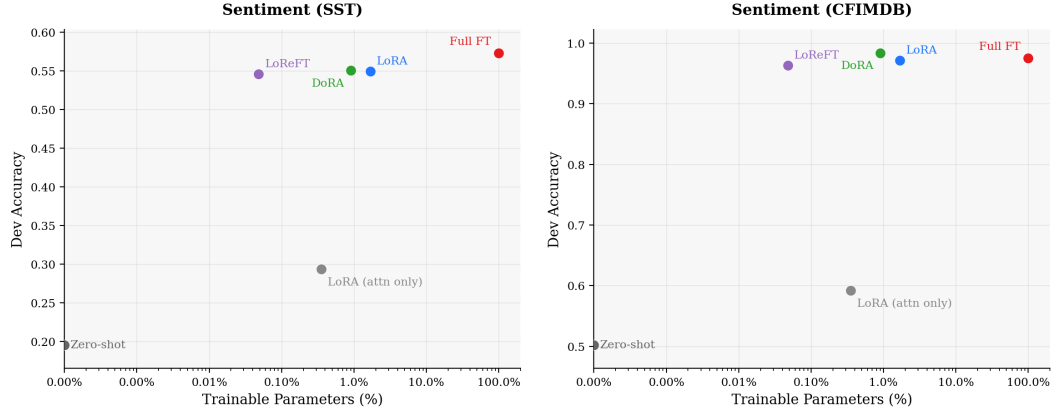
Imogen: Conducted literature review, implemented LoRA and LoReFT, created figures, and wrote final report.

References

- [1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.
- [2] 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.
- [3] 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.
- [4] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: representation finetuning for language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.
- [5] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [6] 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.
- [7] John Schulman. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. <https://thinkingmachines.ai/blog/lora/>.
- [8] Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. Extracting latent steering vectors from pretrained language models. *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.
- [9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.
- [11] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. PEFT: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>, 2022.

A Supplementary Figures

Figure 4: **Parameter efficiency vs. dev performance on GPT-2 Small for sentiment analysis evaluated on SST and CFIMDB.** Each point represents a method’s best dev performance plotted against its trainable parameter percentage (log-scaled). Attention-only LoRA (LoRA[†]) underperforms LoRA applied to all linear layers. Similar plots for paraphrase detection and sonnet generation are given in Table 2.



B Paraphrase Detection Experiment Results

To save on compute, we only tuned a small number of parameters on the Paraphrase Detection task on the base GPT-2 model and extrapolated those tuned parameters to the Sonnet Generation and Sentiment Classification tasks along with larger model sizes.

Table 3: **Tuned hyperparameter configurations for GPT-2 small (124M parameters)**, each trained for 10 epochs and selected by dev set accuracy. Learning rates chosen from sweep over {5e-5, 1e-4, 2e-4, 5e-4}. LoRA/DoRA ranks were chosen from sweep over {2, 4, 8, 16}. All methods use AdamW optimizer, batch size 8, and weight decay 0.0. These configurations are used for all subsequent experiments (i.e., sonnet generation, sentiment analysis, scaling to GPT-2 large). Refer to full sweep results below.

Method	LR	Rank	Target	Positions	Layers	Params (%)	Dev Acc
Full FT	2e-5	—	All weights	—	—	100%	0.897
LoRA★	1e-4	16	c_attn, c_proj, c_fc	—	—	~2.10%	0.8930
DoRA†	1e-4	8	c_attn, c_proj, c_fc	—	—	~1.12%	0.8927
LoReFT	5e-4	4	—	f4+l4	All	~0.06%	0.8533

Table 4: **Baseline results for paraphrase detection.** All runs use GPT-2 small, batch size 8, 10 epochs. We select the best full FT learning rate in the main results table (see Table 2).

Run	LR	Params (%)	Dev Acc	F1
<i>Baselines on GPT-2 Small</i>				
Zero-shot	—	0%	0.3899	0.3326
Full FT	1e-5	100%	0.8958	0.8874
Full FT	2e-5	100%	0.8970	0.8906
Full FT	5e-5	100%	0.8947	0.8878
LoRA (attn only; rank 8)	1e-5	~0.354%	0.8486	0.8404

Table 5: **Paraphrase detection across GPT-2 model sizes.** Each method uses its best LR and rank from GPT-2 small sweeps without re-tuning. Params (%) decreases with model size as adapter sizes stay fixed. Batch size was reduced to 4 (medium) and 2 (large) for GPU memory.

Model	Method	LR	Rank	Params (%)	Dev Acc	F1
<i>GPT-2 Small (124M, 12 layers, d=768)</i>						
Small	Full FT	—	—	100%	0.8970	0.8906
Small	LoRA★	1e-4	16	2.1%	0.8930	0.8869
Small	DoRA★	1e-4	8	1.12%	0.8927	0.8854
Small	LoReFT★	5e-4	4	0.06%	0.8532	0.8444
<i>GPT-2 Medium (355M, 24 layers, d=1024)</i>						
Medium	Full FT	2e-5	—	100%	0.9055	0.8986
Medium	LoRA★	1e-4	16	1.95%	0.9033	0.8972
Medium	DoRA★	1e-4	8	1.05%	0.9016	0.8961
Medium	LoReFT★	5e-4	4	0.06%	0.8827	0.8751
<i>GPT-2 Large (774M, 36 layers, d=1280)</i>						
Large	Full FT	1e-6	—	100%	0.9102	—
Large	LoRA★ (all)	1e-4	16	1.68%	0.9025	0.8959
Large	DoRA★ (all)	1e-4	8	0.90%	0.9054	0.8997
Large	LoReFT★	5e-4	4	0.05%	0.8883	0.8821

C Paraphrase Detection Hyperparameter Sweep Results

Table 6: **Learning rate sweeps for all three PEFT methods on paraphrase detection.** We set LoRA to target all weight matrices following the work of Schulman & Thinking Machines Lab [7]. DoRA uses half the rank of LoRA to approximately match parameter count. LoReFT interventions are applied at first-2 and last-2 token positions across all layers [4] The best LR per method (measured by dev accuracy) is **bolded** and used for all subsequent experiments.

Run	LR	Rank	Params (%)	Dev Acc	F1
<i>Learning rate sweeps for LoRA (rank 8) on all layers</i>					
LoRA	5e-5	8	~1.05%	0.8866	0.8812
LoRA	1e-4	8	~1.05%	0.8919	0.8845
LoRA	2e-4	8	~1.05%	0.8847	0.8760
LoRA	5e-4	8	~1.05%	0.8284	0.8210
<i>Learning rate sweeps for DoRA (rank 4) on all layers</i>					
DoRA	5e-5	4	~0.6%	0.8864	0.8796
DoRA	1e-4	4	~0.6%	0.8909	0.8849
DoRA	2e-4	4	~0.6%	0.8876	0.8812
DoRA	5e-4	4	~0.6%	0.8665	0.8584
<i>Learning rate sweeps for LoReFT (rank 4, all layers, f2+l2)</i>					
LoReFT	5e-5	4	~0.06%	0.8452	0.8366
LoReFT	1e-4	4	~0.06%	0.8498	0.8418
LoReFT	2e-4	4	~0.06%	0.8507	0.8423
LoReFT	5e-4	4	~0.06%	0.8532	0.8444

Table 7: **Phase 3: Rank sensitivity for LoRA $\star$ and DoRA on paraphrase detection.** All runs use lr=1e-4 (the best lr from Phase 2) and target all linear layers. LoRA $\star$ r=8 and DoRA $\star$ r=4 rows are carried over from Phase 2. The best rank per method is **bolded**. Note that at matched ranks (e.g. r=4), DoRA outperforms LoRA (0.8904 vs 0.8899), and at r=2 the gap may even widen.

Run	LR	Rank	Params (%)	Dev Acc	F1
<i>Rank sweeps for LoRA (lr=1e-4) on all layers</i>					
LoRA	1e-4	4	~0.53%	0.8899	0.8839
LoRA	1e-4	8	~1.05%	0.8919	0.8845
LoRA	1e-4	16	~2.10%	0.8930	0.8869
LoRA	1e-4	32	~4.74%	0.8916	0.8851
<i>Rank sweeps for DoRA (lr=1e-4) on all layers</i>					
DoRA	1e-4	2	~0.33%	0.8843	0.8778
DoRA	1e-4	4	~0.59%	0.8904	0.8849
DoRA	1e-4	8	~1.12%	0.8927	0.8854
DoRA	1e-4	16	~2.14%	0.8922	0.8858

Table 8: **Phase 4: LoReFT ablations on paraphrase detection.** Phase 4a varies token positions (all 12 layers) while Phase 4b varies layer subsets (f4+l4 positions). All use rank 4, lr=5e-4. Although f2+l2 were similar in performance, we ended up selecting f4+l4 for its more stable training. Top 6 layers outperform bottom 6, suggesting higher layers encode more task-relevant semantics.

Configuration	LR	Rank	Params (%)	Dev Acc	F1
<i>LoReFT position ablation (all 12 layers, lr=5e-4)</i>					
l1 (last 1 only)	5e-4	4	~0.06%	0.8348	0.8255
f1+l1 (first 1 + last 1)	5e-4	4	~0.06%	0.8523	0.8431
f2+l2 (first 2 + last 2)	5e-4	4	~0.06%	0.8532	0.8444
f4+l4 (first 4 + last 4)	5e-4	4	~0.06%	0.8533	0.8445
<i>LoReFT layer ablation (f4+l4 positions, lr=5e-4)</i>					
All 12 (0–11)	5e-4	4	~0.06%	0.8533	0.8445
Top 3 (9–11)	5e-4	4	~0.015%	0.8093	0.7941
Top 6 (6–11)	5e-4	4	~0.03%	0.8378	0.8303
Bottom 6 (0–5)	5e-4	4	~0.03%	0.8335	0.8277

D Sonnet Generation Experiment Results

Method	Rank	LR	Params (%)	Epochs	Dev chrF
<i>Baselines on GPT-2 Small</i>					
Zero-shot	—	—	0%	—	40.2544
Full FT	—	1e-5	100%	10	41.2052
Full FT	—	2e-5	100%	10	42.1394
Full FT	—	5e-5	100%	10	42.3323
LoRA (attn only)	8	1e-5	0.35%	10	41.4625
<i>PEFT Methods on GPT-2 Small</i>					
LoRA★ (all linear)	16	1e-4	2.07%	20	41.7858
DoRA★ (all linear)	8	1e-4	1.12%	20	41.9569
LoReFT (f4+l4, all layers)	4	5e-4	0.06%	20	27.3816
LoReFT (all pos, all layers)	4	5e-4	0.06%	20	41.7891
<i>Full FT + PEFT Methods on GPT-2 Medium</i>					
Full FT	—	5e-5	100%	20	42.5483
LoRA★ (all linear)	16	1e-4	1.95%	20	42.6296
DoRA★ (all linear)	8	1e-4	1.05%	20	42.2304
LoReFT (all pos, all layers)	4	5e-4	0.02%	20	42.4910
<i>Full FT + PEFT Methods on GPT-2 Large</i>					
Full FT	—	5e-5	100%	20	42.9333
LoRA★ (all linear)	16	1e-4	1.68%	20	42.9150
DoRA★ (all linear)	8	1e-4	0.90%	20	42.9121
LoReFT (all pos, all layers)	4	5e-4	0.01%	20	42.2817

Table 9: **Sonnet generation results across GPT-2 model sizes.** Note baselines use batch size 8 for 10 epochs and PEFT methods train for 20 epochs. GPT-2 Small hyperparameters are selected from paraphrase detection sweeps that are described in Table 3. The Medium and Large models use the same configuration settings without additional tuning. The Batch size was reduced to 4 (Medium) and 2 (Large) to fit in GPU memory. LoRA (attn only) uses the project handout defaults, while the weak result motivates targeting all linear layers (★). Params (%) decreases with model size as adapter sizes stay fixed.

E Sentiment Classification Experiment Results

Method	Rank	LR	Params (%)	SST Dev Acc	CFIMDB Dev Acc
<i>Baselines on GPT-2 Small</i>					
Zero-shot	—	—	0%	0.1953	0.5020
Full FT	—	2e-5	100%	0.5086	0.9755
LoRA (attn only)	8	1e-5	0.35%	0.2934	0.5918
<i>PEFT Methods on GPT-2 Small</i>					
LoRA* (all linear)	16	1e-4	2.08%	0.5068	0.9755
DoRA* (all linear)	8	1e-4	1.12%	0.5059	0.9796
LoReFT (f4+14, all layers)	4	5e-4	0.06%	0.5204	0.9755
LoReFT (all pos, all layers)	4	5e-4	0.06%	0.5186	0.9755
<i>Full FT + PEFT Methods on GPT-2 Medium</i>					
Full FT	—	5e-5	100%	0.5441	0.9714
LoRA* (all linear)	16	1e-4	1.95%	0.5559	0.9796
DoRA* (all linear)	8	1e-4	1.05%	0.5577	0.9674
LoReFT (all pos, all layers)	4	5e-4	0.06%	0.5431	0.9674
<i>Full FT + PEFT Methods on GPT-2 Large</i>					
Full FT	—	5e-5	100%	0.5731	0.9755
LoRA* (all linear)	16	1e-4	1.68%	0.5495	0.9714
DoRA* (all linear)	8	1e-4	0.90%	0.5504	0.9837
LoReFT (all pos, all layers)	4	5e-4	0.05%	0.5459	0.9633

Table 10: **Sentiment classification results across GPT-2 model sizes.** All non-zero-shot runs trained for 10 epochs with batch size 64. SST uses 5-class classification, while CFIMDB uses binary classification. GPT-2 Small hyperparameters are selected from paraphrase detection sweeps (refer to Table 3); Medium and Large transfer the same configs without re-tuning. LoRA (attn only) uses the project handout defaults. Note that the weak result motivates targeting all linear layers (*). Params (%) decreases with model size as adapter sizes stay fixed.