

Parameter-Efficient Fine-Tuning of GPT-2 Using DoRA

Stanford CS224N Default Project

Linika Goel

Department of Radiology
Stanford University
linika@stanford.edu

Mindy Harkness

Department of Radiation Oncology
Stanford University
mharkness@stanford.edu

Abstract

We implement GPT-2 from scratch and demonstrate its capabilities on three core tasks: sentiment classification, paraphrase detection, and Shakespearean sonnet generation. We establish baseline performance by matching or exceeding reference development accuracies for sentiment classification. To improve downstream performance, we implement two parameter-efficient fine-tuning (PEFT) methods, Low-Rank Adaptation (LoRA) and Weight-Decomposed Low-Rank Adaptation (DoRA), and compare them with full fine-tuning. We find that DoRA outperforms LoRA on both tasks, achieving results comparable to full fine-tuning while training only a fraction of the parameters. Beyond task-level metrics, we introduce a magnitude drift ratio diagnostic to track how DoRA’s learned magnitude vectors evolve across layers and epochs. We find that value and attention output projections exhibit the greatest magnitude drift, while FFN down-projections remain the most stable. The paraphrase detection task induces substantially greater magnitude drift than sonnet generation, suggesting that discriminative tasks require more reshaping of pretrained representations. This analysis points toward a more selective DoRA strategy that could reduce adapter overhead while preserving adaptation quality.

1 Key Information to include

- Mentor: Sarah Chen
- Late Days (if you are using any):

Team member	Late days remaining
Linika Goel	6
Mindy Harkness	6
Total pooled	12
Team size	2
Late days applied for the report	2

2 Introduction

Fine-tuning large language models (LLMs) through full fine-tuning (FT) is computationally expensive, as it requires updating all model parameters. Parameter-efficient fine-tuning (PEFT) methods address this by freezing pretrained weights and training only a small set of additional or low-rank parameters, significantly reducing computational cost.

A popular PEFT method is Low-Rank Adaptation (LoRA) [1]. LoRA models the weight update using low-rank decomposition matrices $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$, which are trained while the pretrained weights remain frozen:

$$W' = W_0 + BA \tag{1}$$

where W_0 is the frozen pretrained weight matrix and $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is the rank. While LoRA dramatically reduces the number of trainable parameters, it typically does not fully match the accuracy of full fine-tuning.

Liu et al. [2] investigate the gap between LoRA and FT by decomposing weight updates into *magnitude* and *direction* components. They show that LoRA tends to couple magnitude and directional updates linearly, whereas FT often exhibits a negative correlation between the two. This observation motivates Weight-Decomposed Low-Rank Adaptation (DoRA), which explicitly separates these components. DoRA decomposes the pretrained weight W_0 into a trainable magnitude vector $\mathbf{m} \in \mathbb{R}^{d_{\text{out}}}$ and a directional matrix V . The directional component is updated via standard LoRA, while $\mathbf{m}$ is fully trainable. In the forward pass, DoRA computes:

$$W' = \mathbf{m} \odot \frac{V + \delta V}{\|V + \delta V\|_c} = \mathbf{m} \odot \frac{W_0 + BA}{\|W_0 + BA\|_c} \quad (2)$$

where $\|\cdot\|_c$ denotes column-wise norms. The components are merged before inference, ensuring no additional latency.

The primary goal of this project is to build a GPT-2 model following the standard transformer architecture, implement both LoRA and DoRA, and evaluate how well these PEFT methods adapt GPT-2 to downstream tasks, paraphrase detection and sonnet generation, compared to full fine-tuning. Beyond comparing task accuracy, we introduce a novel diagnostic: the *magnitude drift ratio*, which tracks how DoRA’s magnitude vectors evolve across layers and epochs, revealing which parts of the transformer benefit most from the magnitude–direction decomposition.

3 Related Work

Parameter-efficient fine-tuning (PEFT) methods enable adaptation of large pretrained language models by updating only a small subset of parameters while keeping the backbone weights frozen. Early work explored incorporating lightweight adapter modules into transformer layers for task-specific adaptation [3]. Prompt-based approaches, such as prefix tuning and prompt tuning, instead optimize continuous prompts that condition the model without modifying its internal weights [4, 5].

Low-rank adaptation methods have emerged as a particularly effective PEFT paradigm. LoRA [1] has become a widely used baseline across NLP tasks, and QLoRA [6] extends it by combining low-rank adaptation with quantization to further reduce memory requirements. Additional work has explored adaptive rank strategies, including AdaLoRA [7] and AG-LoRA [8]. Despite their efficiency, these methods do not fully close the gap with full fine-tuning, and prior work has generally attributed this to the limited number of trainable parameters.

DoRA [2] challenges this explanation by decomposing LoRA’s weight updates into magnitude and direction. Analysis of these two components reveals that LoRA updates these two components linearly, whereas full fine-tuning often exhibits a *negative* correlation between magnitude and directional adjustments, creating a richer learning dynamic that LoRA’s parameterization cannot express. DoRA restores this negative correlation by explicitly separating the magnitude and directional components: a trainable scalar vector $\mathbf{m}$ controls per-output scaling, while a standard LoRA update governs directionality. Across common sense reasoning benchmarks, DoRA improves over LoRA by 3–4%, with these gains most pronounced when the complexity gap between pretraining and the downstream task is large.

Our work directly builds on this analysis. The DoRA paper tracks the magnitude of individual layer weight vectors over training steps to motivate its decomposition. It does not analyze how this magnitude drifts by sublayer type (Q , K , V , Attn , FFN_1 , FFN_2) or by layer depth. We introduce the magnitude drift ratio (Eq. 4) and provide a systematic analysis of how DoRA’s learned magnitude vectors evolve across both sublayer function and layer depth, revealing a consistent sublayer hierarchy and a task-dependent depth gradient that has practical implications for more selective fine-tuning strategies.

4 Approach

4.1 GPT-2 Architecture

GPT-2 [9] is a decoder-only Transformer trained with an autoregressive language modeling objective. Our implementation follows the standard GPT-2 architecture: token and positional embeddings, L masked multi-head self-attention layers, feed-forward networks with GELU activation, and residual connections with layer normalization, built from scratch using the CS224N default project framework. For our base model we use GPT-2 Small ($d = 768$, $L = 12$, $H = 12$ attention heads, ~ 117 M parameters). For sonnet generation we additionally evaluate GPT-2 Medium ($d = 1024$, $L = 24$).

4.2 LoRA

For LoRA [1], we replace each target linear layer with a frozen weight W_0 plus a trainable low-rank update:

$$h = W_0 x + \frac{\alpha}{r} B A x \quad (3)$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$, α is a scaling hyperparameter, and A is initialized with Kaiming uniform while B is initialized to zero so the adapter begins as an identity. We apply LoRA to all six linear projections in each transformer block: Q , K , V , attention output, FFN up-projection, and FFN down-projection.

4.3 DoRA

For DoRA [2], each adapted layer decomposes the weight into magnitude and direction using Equation 2. The magnitude vector $\mathbf{m} \in \mathbb{R}^{d_{\text{out}}}$ is initialized to the column-wise norms of W_0 and is fully trainable. The directional component is updated via low-rank matrices B and A as in LoRA. DoRA is applied to the same six projection types as LoRA.

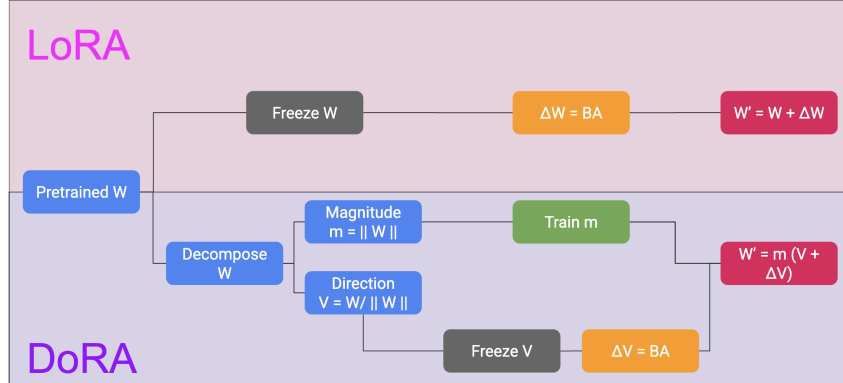


Figure 1: Flowchart of LoRA and DoRA PEFT methods. LoRA and DoRA both begin with pretrained W , then diverge.

4.4 Full Fine-Tuning

As an upper-bound baseline for all downstream tasks, we perform full fine-tuning (FT), unfreezing all parameters of the pretrained GPT-2 model and training with the same optimizer as the PEFT runs.

4.5 Magnitude Drift Analysis for Weight-Decomposed Adapters

To characterize how DoRA’s learned magnitude vectors evolve during fine-tuning, we introduce the *magnitude drift ratio*, a per-layer diagnostic measuring the relative displacement of the magnitude component from its initialization.

For adapter layer ℓ at training step t , the magnitude drift ratio is:

$$\rho_{\ell}^{(t)} = \frac{\|\mathbf{m}_{\ell}^{(t)} - \mathbf{m}_{\ell}^{(0)}\|}{\|\mathbf{m}_{\ell}^{(0)}\| + \epsilon} \quad (4)$$

where $\epsilon = 10^{-12}$ ensures numerical stability. This quantity is zero at initialization and grows as the magnitude vector departs from its pretrained scaling. Unlike the full weight drift $\|\Delta W\|/\|W\|$, which conflates magnitude and directional changes, $\rho_{\ell}^{(t)}$ isolates the magnitude component, directly measuring how much DoRA’s decomposition is leveraged at each layer.

We report ρ disaggregated by sublayer function within each block. In a standard GPT-2 block, DoRA adapts six projections: Q , K , V , attention output (Attn), FFN up-projection (FFN₁), and FFN down-projection (FFN₂). We compute the mean drift at the final training step:

$$\bar{\rho}_s = \frac{1}{L} \sum_{\ell \in \mathcal{L}_s} \rho_{\ell}^{(T)} \quad (5)$$

where $\mathcal{L}_s$ is the set of layers of sublayer type s and T is the final step.

We further aggregate drift by transformer block index $n \in \{0, \dots, N-1\}$:

$$\bar{\rho}_n = \frac{1}{6} \sum_{\ell \in \text{block}(n)} \rho_{\ell}^{(T)} \quad (6)$$

capturing whether the task requires more adaptation at early layers (near the embedding) or late layers (near the output).

5 Experiments

5.1 Data

All experiments use the datasets provided for the CS224N default project:

- **SST** [10]: 5-class sentiment classification. Train / dev / test: 8,544 / 1,101 / 2,210.
- **CFIMDB**: Binary sentiment classification. Train / dev / test: 1,701 / 245 / 488.
- **Quora Question Pairs (QQP)**: Binary paraphrase detection. Train / dev / test: 141,506 / 20,215 / 40,431.
- **Shakespeare Sonnets**: Autoregressive generation. Train / test: 143 / 12 sonnets.

All inputs are tokenized using GPT-2 byte-pair encoding. For classification tasks, we extract the final-token hidden state and pass it through a linear head.

5.2 Evaluation

Classification tasks are evaluated via accuracy. For sonnet generation, we employ chrF [11], a character-level n-gram metric. We also track training loss convergence and the train-dev accuracy gap to assess overfitting behavior. For DoRA runs, we additionally log the magnitude drift ratio (Eq. 4) every 50 training steps.

5.3 Baselines

For sentiment classification, we compare against the reference baselines provided in the CS224N project guidelines. For paraphrase detection and sonnet generation, the full fine-tuning run serves as the upper-bound baseline.

5.4 Sentiment Classification

To validate our GPT-2 implementation, we evaluate two configurations: a last-layer-only classifier head, and full fine-tuning. For full fine-tuning, we use a learning rate of $1e-5$, batch size of 8, and train for 10 epochs. Table 1 shows that our model matches or exceeds the reference baseline in 3 of 4

settings, confirming a correct implementation. While we observe a slight shortfall on SST last-layer tuning (−1.8 pts), this is likely due to high variance on the noisy 5-class task. FT on SST shows overfitting after epoch 5 (train 0.795 vs. dev 0.495), providing further motivation for the parameter constraints offered by LoRA and DoRA.

Dataset	Method	Our Dev Acc	Baseline Dev Acc	Diff
SST	Last Linear Layer	0.444	0.462	−0.018
	Full Fine-Tuning	0.520	0.513	+0.007
CFIMDB	Last Linear Layer	0.861	0.861	0.000
	Full Fine-Tuning	0.980	0.976	+0.004

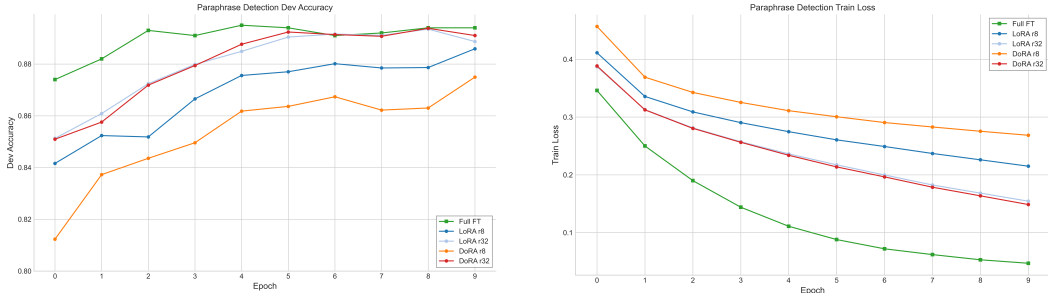
Table 1: Sentiment classification dev accuracy. Our model matches or exceeds the baseline in 3 of 4 settings.

5.5 Paraphrase Detection

We evaluate paraphrase detection with our GPT-2 model across multiple PEFT configurations and full fine-tuning, allowing us to study the effect of rank and PEFT choice on dev accuracy and loss. All evaluations use a batch size of 8. See Table 2 for results of these experiments. Additionally, the rank 32 DoRA test set evaluation is submitted to the leaderboard, with a test accuracy of 0.870.

Method	Rank	α	LR	Trainable %	Best Dev Acc	Final Loss
Full FT	—	—	1e−5	100%	0.8950	0.0470
LoRA	8	16	2e−5	3.3%	0.8859	0.2150
LoRA	32	32	5e−5	11.8%	0.8935	0.1544
DoRA	8	16	2e−5	3.5%	0.8749	0.2685
DoRA	32	32	5e−5	12.0%	0.8938	0.1487

Table 2: Paraphrase detection on GPT-2 Small. Trainable % denotes the fraction of parameters updated during training.



(a) Development-set accuracy across training epochs for paraphrase detection using GPT-2.

(b) Training loss across epochs for the paraphrase detection task under the same fine-tuning configurations.

Figure 2: Training dynamics for paraphrase detection using GPT-2. Accuracy and loss are tracked across epochs for full fine-tuning, LoRA, and DoRA configurations.

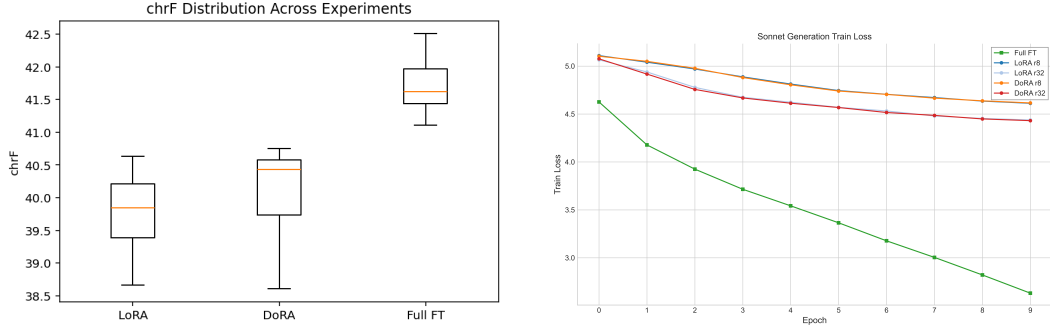
See above for the development-set accuracy and training loss across epochs for all paraphrase detection experiments.

5.6 Sonnet Generation

Table 3 summarizes the experimental configurations we explore. Figure 3 shows the resulting chrF score distributions.

Experiment	Settings
Baseline	GPT-2; batch size 4; 10 epochs; $lr = 5 \times 10^{-5}$; $T = 0.9$; $p = 0.9$
Decoding sweep	$T \in \{0.8, 0.85, 0.9, 0.95\}$; $p \in \{0.88, 0.9, 0.92\}$
LR sweep	$lr \in \{2, 3, 4, 5, 6\} \times 10^{-5}$
Adapter sweep	(LoRA/DoRA) $r \in \{8, 16, 32\}$; $\alpha \in \{16, 32\}$; dropout $\in \{0, 0.05\}$
Model size	GPT-2-medium; batch size 2

Table 3: Experimental configurations for GPT-2 sonnet generation across LoRA, DoRA, and full fine-tuning. All runs use 10 training epochs; decoding parameters (T , p) denote temperature and top- p sampling respectively. Adapter hyperparameters apply only to LoRA and DoRA.



(a) Distribution of chrF scores across fine-tuning methods for sonnet generation. Each box plot summarizes results from multiple experimental runs, showing the median, interquartile range, and overall variability.

(b) Training loss curves for sonnet generation across epochs. The plot compares optimization dynamics for full fine-tuning, LoRA, and DoRA configurations.

Figure 3: Evaluation of fine-tuning strategies for sonnet generation using GPT-2.

5.7 Leaderboard

Task	Method	Dev	Test
Paraphrase Detection	DoRA	0.894 (Acc.)	0.87 (Acc.)
Sonnet Generation	DoRA	42.461 (chrF)	40.2 (chrF)
Sonnet Generation	Full FT	42.509 (chrF)	42.317 (chrF)

6 Analysis

6.1 LoRA vs. DoRA vs. Full Fine-Tuning

Paraphrase Detection. As shown in Table 2, higher-rank adapters substantially improve performance for both PEFT methods. At rank 32, DoRA and LoRA yield comparable results to full FT, with a -0.13% versus -0.17% accuracy difference for DoRA and LoRA respectively. This makes sense given the discussion of DoRA in the original paper [2], which states that DoRA’s gains over LoRA are most pronounced when the complexity gap between pretraining and the downstream task is large. For this paraphrase detection task, there is no complexity gap between pretraining and the downstream task, so a large gap between LoRA and DoRA would not be expected. DoRA and LoRA have similar training loss dynamics (see Figure 2), with both PEFT methods leading to a higher loss than full FT with similar accuracy. This suggests the model does not need full fine-tuning’s extra capacity to learn this task efficiently.

Sonnet Generation. Figure 3 shows the distribution of chrF scores across nine hyperparameter configurations per method for sonnet generation. On the development set, our best DoRA model achieved 42.461 chrF, while full fine-tuning (FT) achieved 42.509. On the test leaderboard, full FT achieved 42.317 chrF (our final leaderboard score) and DoRA achieved approximately 40.2 chrF. Although dev scores were similar, the larger drop for DoRA suggests that dev-tuned configurations did not fully generalize to unseen sonnets, likely due to the small dataset and extensive hyperparameter

tuning. Among PEFT methods, DoRA consistently outperforms LoRA while maintaining similar variance across runs. The performance difference between LoRA and DoRA is greater for the sonnet generation task than the paraphrase detection task. This can be understood by following the same complexity analysis as above; the generative sonnet task has a greater complexity gap between pretraining and the downstream task, contributing to DoRA’s sizable improvement over LoRA. Experiments varying the adapter rank (8, 16, 32) show only small performance differences, suggesting low-rank adapters are sufficient for this generation task.

6.2 Magnitude Drift

Both paraphrase detection and sonnet generation reveal a consistent sublayer hierarchy in magnitude drift (Table 4).

Sublayer	Paraphrase $\bar{\rho}_s$	Sonnet $\bar{\rho}_s$	Ratio (P/S)	Role
V (value)	0.01352	0.000531	$25.5\times$	Highest drift
Attn (output proj)	0.01256	0.000516	$24.4\times$	High drift
K (key)	0.00998	0.000422	$23.7\times$	Moderate
Q (query)	0.00954	0.000403	$23.7\times$	Moderate
FFN ₁ (up proj)	0.00964	0.000451	$21.4\times$	Moderate
FFN ₂ (down proj)	0.00641	0.000272	$23.5\times$	Lowest drift

Table 4: Mean magnitude drift ratio $\bar{\rho}_s$ by sublayer type at the final epoch, with the paraphrase-to-sonnet ratio. The structural ordering is consistent across tasks despite a $\sim 25\times$ scale difference.

V and Attn projections exhibit the greatest magnitude drift across both tasks while FFN₂ drifts the least, a hierarchy preserved with Pearson $r = 0.81$ despite the $\sim 25\times$ scale difference. This identifies V and Attn as the most task-sensitive components and the primary sites where DoRA’s magnitude–direction decomposition provides the most benefit, while FFN₂’s large initial magnitudes (~ 10.8) and minimal drift suggest its pretrained scaling generalizes well across tasks.

The direction of the magnitude depth gradient is task-dependent (Figure 4): paraphrase detection shows an early-layer bias (layers 0–5 mean $\bar{\rho}_n \approx 0.011$ vs. 0.009 for layers 6–11, with L0. V the single highest-drifting parameter at $\rho = 0.0278$), consistent with discriminative classification requiring low-level representational reshaping away from GPT-2’s pretraining objective. Sonnet generation shows the opposite, a late-layer bias (layers 10–11 mean $\bar{\rho}_n \approx 0.000475$ –0.000489 vs. ≈ 0.000362 –0.000385 for layers 0–2), consistent with generation tasks relying on higher-level sequence planning at later layers.

As shown in Figure 4, the $\sim 25\times$ difference in absolute drift scale (paraphrase $\bar{\rho}^{(T)} = 0.0103$; sonnet = 0.000433) mirrors this task-type divide, as paraphrase detection demands substantial representational reshaping away from GPT-2’s pretraining objective, while sonnet generation remains close to it. Dataset size may also contribute (QQP’s $\sim 141\text{K}$ examples versus 143 sonnets) but task type is likely the dominant factor.

6.3 Implications for Efficient DoRA

Our magnitude drift analysis motivates a *selective* DoRA strategy. Since FFN₂ is the lowest-drift sublayer across both tasks, removing DoRA there cuts $\sim 17\%$ of magnitude parameters with minimal performance loss. Conversely, V and Attn projections, which drift 1.4 – $2\times$ more than other sublayers, should be prioritized under a constrained parameter budget. Layer selection can further be made task-dependent: early layers (0–5) for discriminative tasks, later layers (8–11) for generation. Together, these principles motivate a selective DoRA configuration: apply full DoRA only to V and Attn projections and plain LoRA to Q , K , FFN₁, and FFN₂, reducing magnitude parameter overhead while preserving adaptation capacity where it matters most.

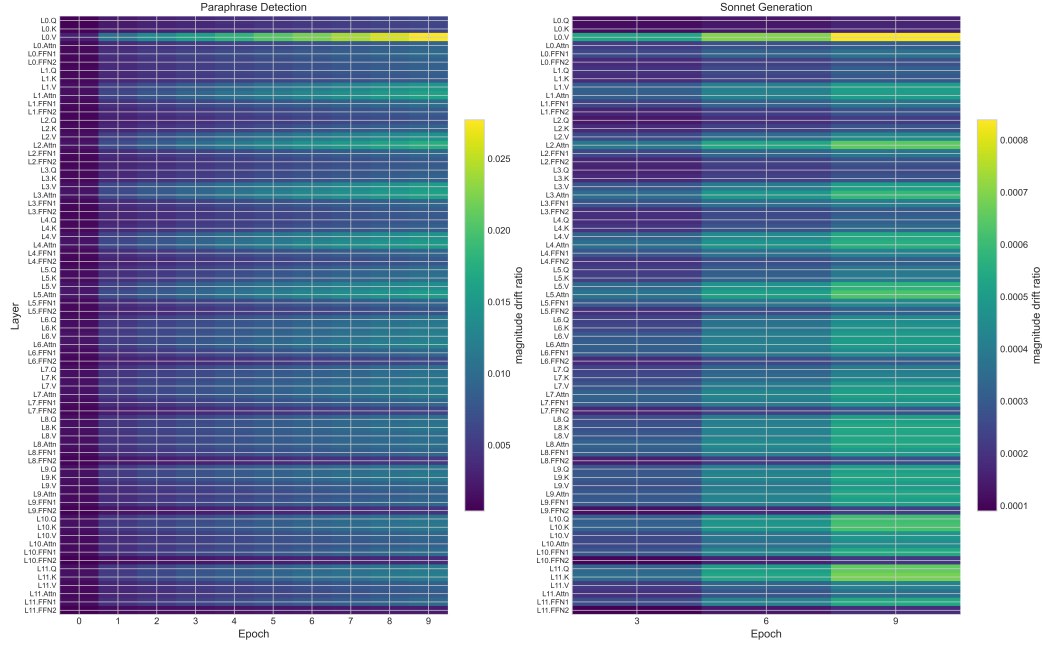


Figure 4: Magnitude vector drift by sublayer and epoch for paraphrase detection (left) and sonnet generation (right) with rank 32 DoRA.

7 Conclusion

We implement GPT-2 from scratch and validate it on sentiment classification, matching or exceeding reference baselines in 3 of 4 settings. For paraphrase detection and sonnet generation, DoRA matches full fine-tuning performance while training only $\sim 12\%$ of parameters, and consistently outperforms LoRA across both tasks. Our magnitude drift analysis reveals a consistent sublayer hierarchy, with V and Attn projections drifting most and FFN_2 drifting least, and a task-dependent depth gradient, with discriminative tasks favoring early-layer adaptation and generation tasks favoring later layers. These findings motivate an efficient DoRA configuration that could reduce magnitude parameter overhead with minimal performance loss. Primary limitations include the small sonnet dataset (143 examples), which constrains generalization conclusions, and the use of small GPT-2 for paraphrase experiments. Further experimentation using larger GPT-2 models on all tasks would help validate our findings. Future work should validate the selective DoRA hypothesis empirically across larger models and more diverse tasks.

Team contributions

Mindy: Attention and GPT-2 layer base. DoRA and LoRA implementation (hand-coded rather than calling a PEFT module). DoRA magnitude drift definition, analysis and heatmap plotting. Paraphrase detection implementation and experiments. Accuracy and loss curve plotting.

Linika: AdamW optimizer implementation. gpt2.py model implementation. classifier.py sentiment pipeline. Sonnet generation implementation and experiments. Hyperparameter sweeps and evaluation.

References

- [1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, and Shean Wang. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [2] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, and Yu-Chiang Frank Wang. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2022.

Learning (ICML), 2024.

- [3] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, et al. Parameter-efficient transfer learning for nlp. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.
- [4] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.
- [5] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.
- [6] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [7] Qingru Zhang, Min Chen, et al. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023.
- [8] Qingchen Wang and Sheng Shen. Activation-guided low-rank parameter adaptation for efficient model fine-tuning. *arXiv preprint arXiv:2501.XXXX*, 2025.
- [9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.
- [10] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics.
- [11] Maja Popović. chrF: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, 2015. Association for Computational Linguistics.

A Appendix

Run	LoRA chrF	DoRA chrF	Full FT chrF
1	39.018	40.581	41.968
2	39.385	38.605	41.110
3	40.207	40.432	41.439
4	40.633	39.506	41.815
5	39.846	40.753	42.267
6	38.666	39.734	41.460
7	40.327	40.347	41.621
8	39.524	40.484	41.165
9	39.935	40.578	42.509

Table 5: chrF scores from nine sonnet generation experiments across LoRA, DoRA, and full fine-tuning (Full FT). These values were used to construct the box plot in Figure 3 showing the distribution of chrF performance across fine-tuning strategies.